

AWS – Compute

Amazon EC2

Amazon Machine Images (AMIs) are preconfigured templates for your instances. Every time you launch an instance from an AMI, a root storage device is created that contains all the information necessary to boot the instance.

Amazon EC2 Auto Scaling offers horizontal scaling of EC2 instances.

Auto Scaling Policy Types:

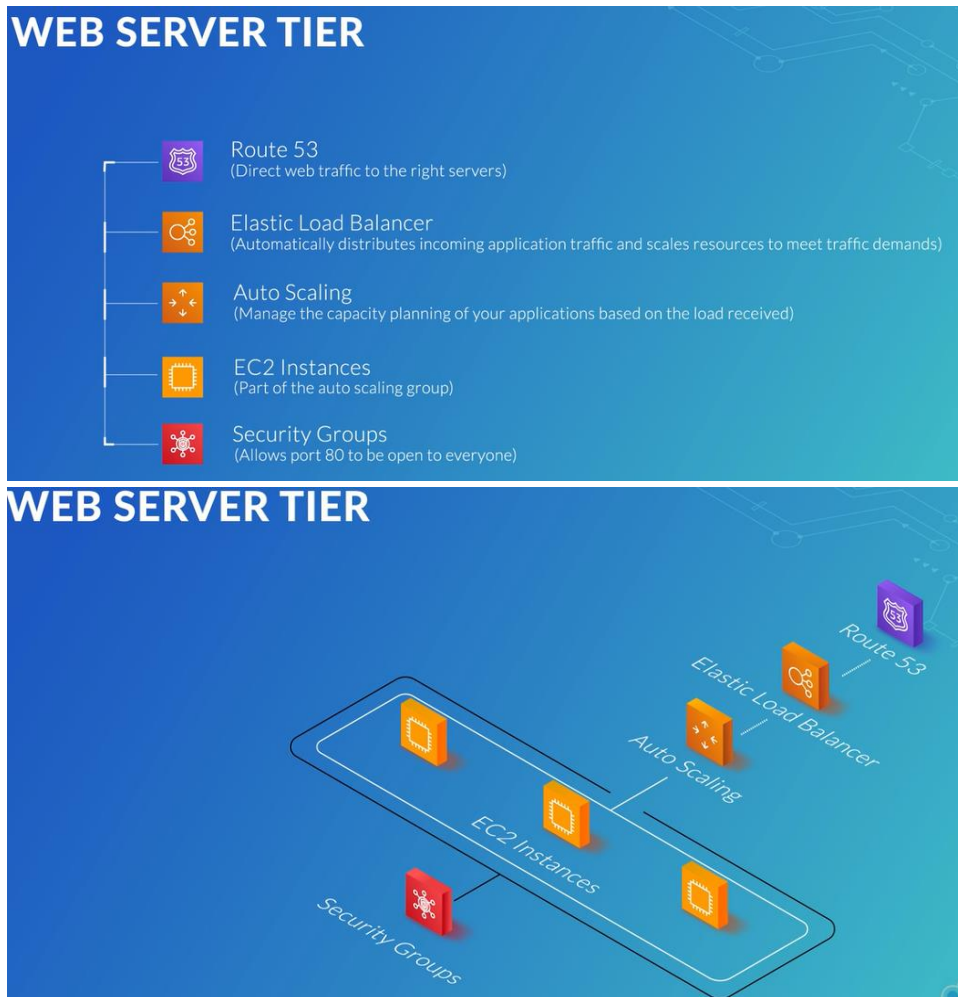
1. Manual Scaling: You must manually add or remove instances based on traffic.
2. Schedule scaling: Scheduled scaling is a great way to help reduce costs because you add or remove instances from your auto scaling groups based on certain time parameters.
3. Dynamic scaling: It removes the burden of having to manually launch instances.
 - 3.1. Step Scaling: Step scaling is a method of adding or removing instances from your autoscaling group based on tracking of a certain metric such as CPU usage. For example: CloudWatch Alarm checks the CPU Usage and deploys 1 instance at 60% CPU Usage, deploys 3 instances at 80% CPU Usage, deploys 5 instances at 95% CPU Usage.
 - 3.2. Target Tracking: Target tracking is a more simplified version of step scaling. You just need to tell the system about where you want the CPU utilization to sit, and it will do the rest.
4. Predictive scaling: This service uses machine learning to understand your workloads. It can learn when your traffic normally rises and falls throughout the day. Based on that knowledge it will provision new instances just before they are needed and will start to get rid of them as traffic trails off.

AWS Elastic Beanstalk

A configuration template is a baseline for creating a new, unique environment using Elastic Beanstalk.

Environment Tiers:

1. Web Server Tier - The webserver environment is associated with HTTP requests.



2. Worker Tier - The worker environment applies mainly to backend processes.



Apart from the environment tiers, if you want your elastic beanstalk service to have additional customization you can do that by using JSON and YAML files.

Deployment Options:

1. All at once: It's the default option. It will simply roll out the application to your resources all at the same time. This will cause a disruption to your application while the update is in progress, which would affect your end users.
2. Rolling: A rolling deployment will deploy your application in batches which will minimize the amount of disruption. This would mean that you have two different versions of the application running at the same time for a very short period. However, it also means that you can still serve requests and process information through your application while the deployment is gradually rolled out for your new infrastructure.
3. Rolling with additional batch: Environment is updated in batches until all your resources have the new update. Adds another batch of instances with your environment to your resource pool to ensure application availability is not impacted.
4. Immutable: Creates an entirely new set of instances. These new instances will be served through a temporary autoscaling group behind your elastic load balancer. This means for a short period of time your environment would essentially double in size. However, once your new instances are deployed and have passed all the health checks the old environment will be removed and the autoscaling group will be updated. If the health checks fail, then the traffic will be served to your original environment.

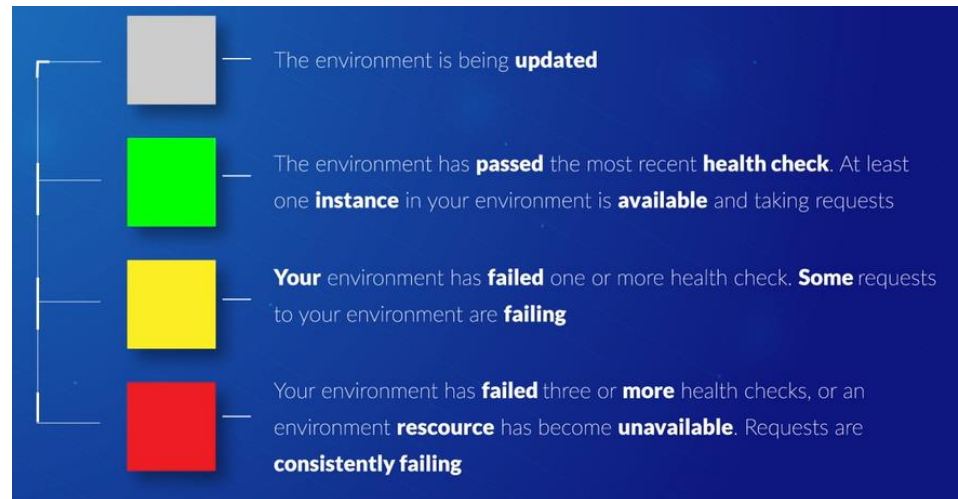
Monitoring and Health Checks:

1. Basic health reporting: Provides an overview of how your environment is performing. Any resources running in your environment will send metrics to Amazon Cloudwatch in five-minute intervals. If you are running a Web Tier, it would include metrics on EC2 or ELB request rates. If you were running in a Worker Tier, then you would have metrics associated to SQS. There are two types of status checks.

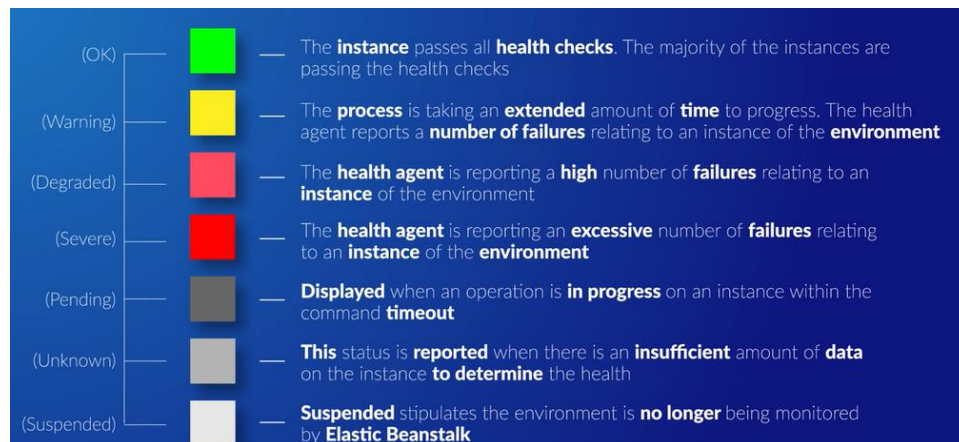
Firstly, the system status check. If this fails, then it's likely to be an issue with the underlying host. Common issues for system status checks to fail are loss of power, loss of network connectivity, and hardware and software issues on the underlying

host. Basically, a system status check failure is out of our control, but the fault lies within AWS resources. The best way to resolve this would be to stop the instance and restart it.

Secondly, we have the instance status check. If this fails, we need to check at the EC2 instance itself. Common issues that trigger this check to fail are incorrect network configuration, corrupt file systems, exhausted memory, or an incompatible kernel. These faults will require you to troubleshoot and resolve the issue. For example, changing the network configuration.



2. Enhanced health reporting: Provides granular level information about the environment that's running. The Elastic Beanstalk health agent reports metrics to Elastic Beanstalk every ten seconds.



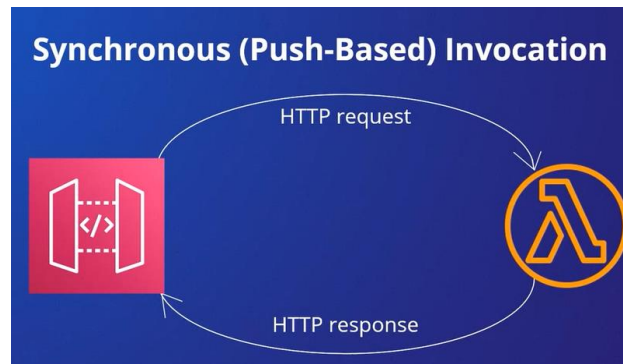
The Elastic Beanstalk health agent is best described as a daemon process that runs on each EC2 instance in your environment.

AWS Lambda

The Lambda handler is the function name inside the code that you write.

Invocation Methods:

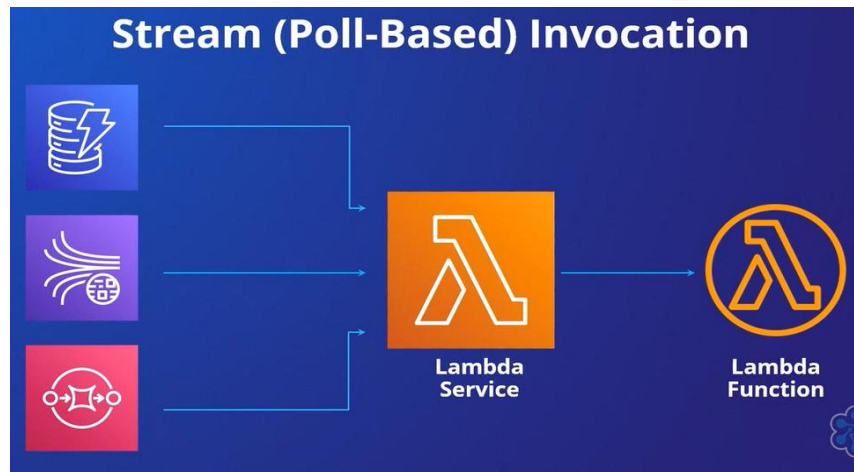
1. Synchronous or push-based model – If your application needs a response.



2. Asynchronous model - The response does not go back to the original service, unless you write that logic yourself. It uses a built-in queue to process events sent to your function.



3. Stream model or poll-based model - This is typically used when you need to poll messages out of stream or queue-based services such as DynamoDB streams, Kinesis streams, and Amazon SQS. The Lambda service runs a poll-er on your behalf, and consumes the data that comes out of them, filtering through them to invoke your Lambda function on only messages that match your use case.



Metrics:

1. Invocation metrics
2. Performance metrics
3. Concurrency metrics

AWS Serverless Application Model (AWS SAM)

AWS SAM has two components:

1. AWS SAM Templates - An AWS SAM template is a configuration file that allows you to define the infrastructure and resources needed for a serverless application in AWS. It provides a simple syntax to describe the functions, APIs, permissions, configurations, and events that make up a serverless application. AWS SAM templates can be written in either YAML or JSON format.
2. AWS SAM Command Line Interface (SAM CLI) -

AWS SAM CLI

- Local development and testing
- Comprehensive commands
- Seamless deployment

AWS SAM Benefits:

1. **Simplicity:** AWS SAM templates simplify the creation of serverless applications.
2. **Integration with Development Tools:** Integrates with IDEs like PyCharm and VS Code. Also, it integrates with AWS serverless tools.
3. **Local Debugging and testing:** Provides an execution environment on your local machine.
4. **Extension of CloudFormation:** You can use Cloudformation's features with SAM templates.
5. **Single Deployment Configuration:** Unified deployment, versioned deployment unit and shared configurations.

AWS – Storage

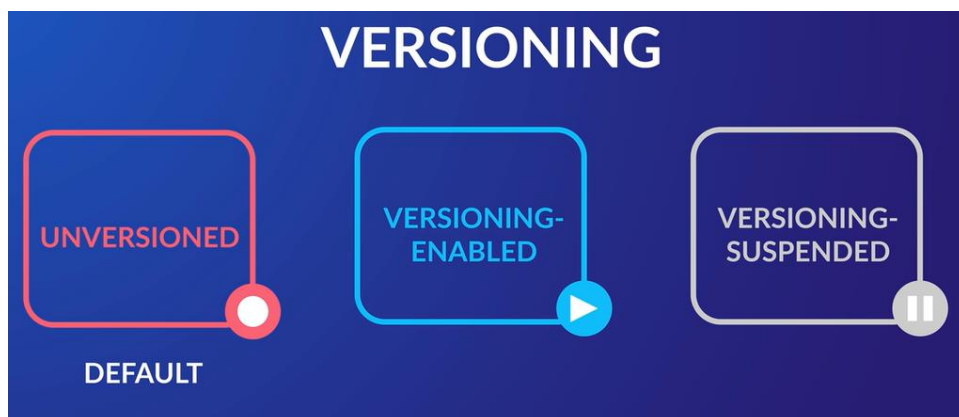
Amazon S3

Each object inside the S3 bucket has exactly one key, which is a unique identifier of the object. An object is uniquely identified with the bucket, key, and version ID.

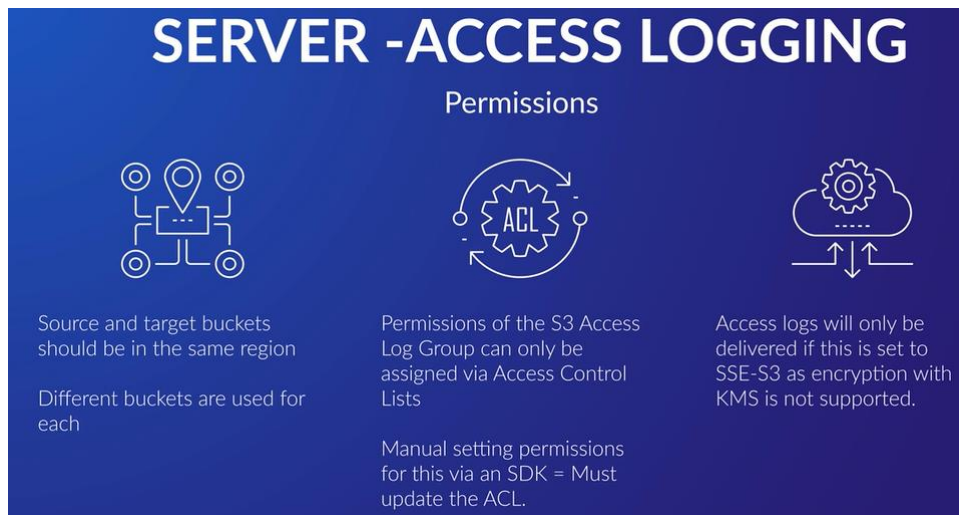
A user can upload objects of size greater than 5 GB to Amazon S3 using multipart upload API.

Versioning - It allows for multiple versions of the same object to exist. This can be useful when a file is accidentally deleted or wants to retrieve previous versions of a file.

Types of Versioning:



Server-Access Logging: It is enabled on a bucket to capture details of requests.



Object-Level Logging: AWS CloudTrail performs logging activities such as capturing API requests (GetObject, DeleteObject, and PutObject) and store it in Amazon S3.

Transfer Acceleration: When you want to transfer data between your Amazon S3 and EC2 instance for example, transfer acceleration can speed up the process by utilizing Amazon CloudFront that uses edge locations.

Bucket Policies:

When working with bucket policies, you must specify a principal element who is associated with the action and effect defined in the statement.

S3 Access Control List - Allows you to control access to buckets and specific objects within a bucket by groupings and AWS accounts.


A canned ACL is a predefined grant that contains both grantees and permissions.

Server-Side Encryption Mechanisms with S3:

1. Server-side encryption with S3 managed Keys (SSE-S3):



2. Server-side encryption with KMS managed keys (SSE-KMS):




Server-side encryption with **KMS managed keys (SSE-KMS)**

- Allows Amazon S3 to use KMS to generate data encryption keys using a KMS key
- You can either use:
 - An **AWS managed** KMS key, which means tasks such as rotation are automatically managed
 - A **customer managed** KMS key, which enables you to disable, rotate, and apply access controls to the KMS key using key policies
- Usage of the key can be tracked and audited using AWS CloudTrail

Using bucket keys reduces the number of requests to KMS and therefore reduces your encryption costs up to as much as 99%.

3. Server-side encryption with Customer Provided Keys (SSE-C):



Server-side encryption with **Customer Provided Keys (SSE-C)**

- Gives you the opportunity to provide your own encryption keys
- Your customer provided key would be sent with your data to S3, which would then perform the encryption for you

Lifecycle configurations for S3:

Lifecycle configurations are an important cost tool that enable you to delete old unused versions of your objects, clean up incomplete multipart uploads, transition objects to lower cost storage tiers and delete objects that are no longer needed.

Components of Lifecycle configurations:

Each lifecycle configuration contains a set of rules. Each rule is broken up into four components: ID (identifies the lifecycle rule), Filters (defines WHICH objects in your bucket you'd like to take action on), Status (you can enable and disable each lifecycle rule), and Actions (define WHERE you want your objects to move to).

Considerations and Limitations for S3:



- Storage transition costs - You get charged when you move data to other storage classes and this fee increases as you move down the staircase. For example, at the top of the staircase, you're charged \$0.01 for every 1,000 lifecycle transition requests when objects are moved from S3 Standard to the S3 Standard-IA storage class. As you go down the staircase, all the way to S3 Glacier Deep Archive, this cost increases, and can be up to \$0.05 for every 1000 transition requests.
- Minimum storage duration periods increase as you go down the staircase as well. For example, S3 Standard and S3 Intelligent-Tiering have no minimum storage duration. Infrequent access tiers like S3 Standard-IA and S3 One Zone - IA have a minimum storage duration of 30 days. Archival storage tiers like S3 Glacier Instant Retrieval and S3 Glacier Flexible Retrieval have a minimum storage duration of 90 days, while S3 Glacier Deep Archive has a minimum storage duration of 180 days. If you delete or overwrite these objects before the minimum storage duration is reached, you get charged. For example, say you transition an object into S3 Glacier Deep Archive for 30 days, and then delete it. In this case, you will still be charged for the full 180 days of storage.

Amazon Elastic File System (EFS)

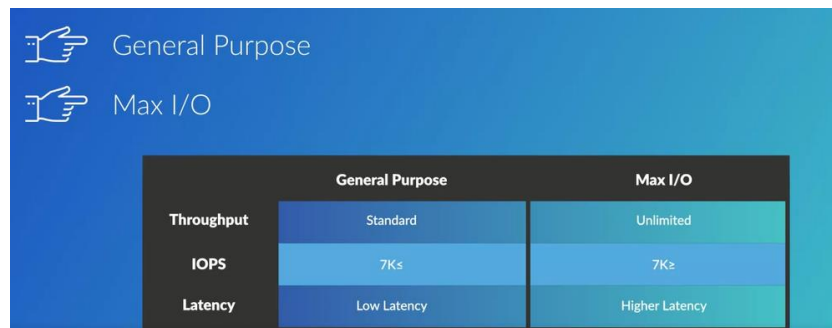
Storage Classes:



| | Standard | IA |
|-------------|------------------|----------------|
| Access | Anytime | Infrequent |
| Cost | Standard Cost | Reduced |
| Performance | Standard Latency | Higher Latency |

EFS IA is cheaper than standard storage. EFS IA also charges for read and write. EFS standard class charges on the amount of storage used each month.

Performance Modes:



| | General Purpose | Max I/O |
|------------|-----------------|----------------|
| Throughput | Standard | Unlimited |
| IOPS | 7Ks | 7Ks+ |
| Latency | Low Latency | Higher Latency |

Throughput Modes:

Bursting Throughput (Default) and Provisioned Throughput.

Block vs Object vs File Storage

Block vs. Object vs. File Storage



Block storage: hard drives and SANs (Amazon EBS)



Object storage: large, unstructured data (Amazon S3)



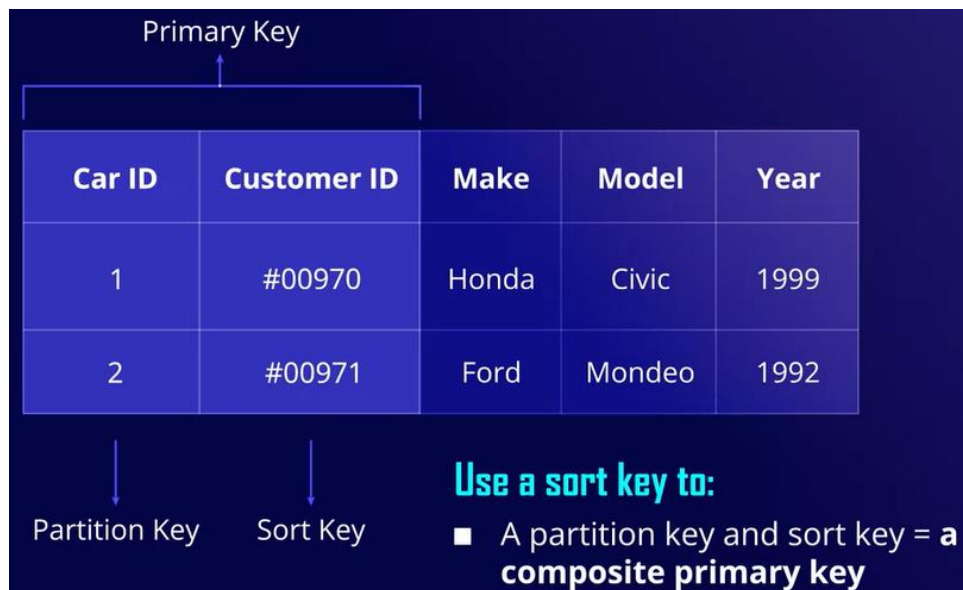
File storage: network-attached storage (Amazon EFS and FSx)

AWS – Databases

Amazon DynamoDB

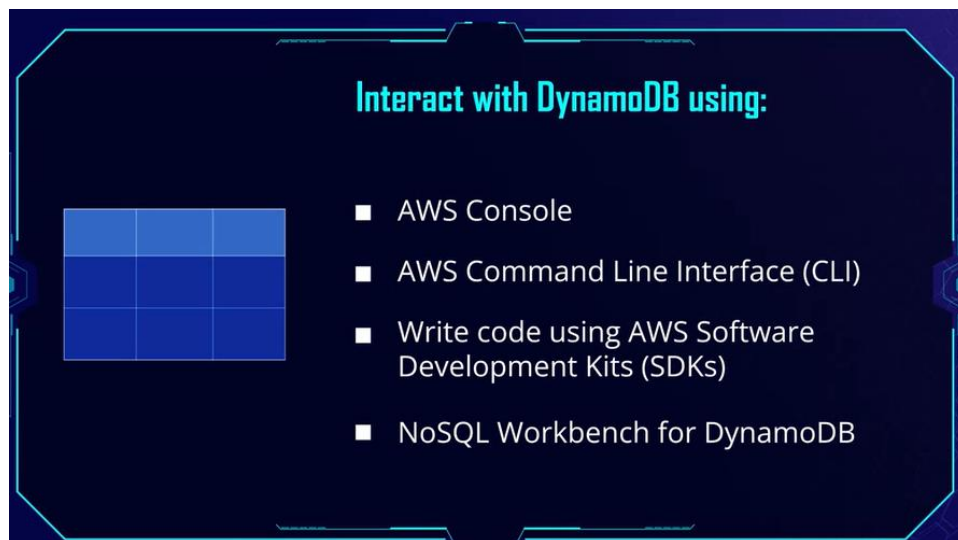


In this example, a partition key or hash key might be a car id or a VIN number. A sort key sorts data within the same partition. For example, you could have a partition key with the car VIN number and a sort key of customer ID. This performs fast querying.

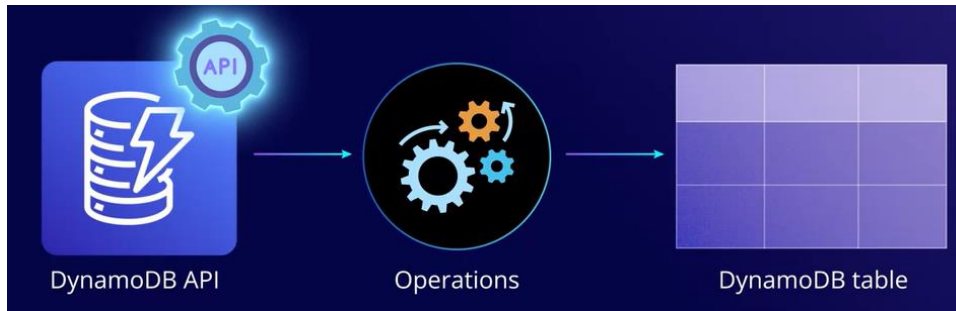


Relational databases use vertical scaling. DynamoDB uses horizontal scaling.

DynamoDB does support PartiQL, which is a SQL-like language for querying and modifying data. DynamoDB supports data types, such as strings, numbers, Boolean values, binary data, null values, lists, maps, and sets. However, if you work with data types like dates, you'll need to represent those as strings or numbers to store them in DynamoDB.




You can use any of the above methods to access the DynamoDB application programming interface or API. The DynamoDB API is organized as a set of operations that you can execute on your DynamoDB tables.



There are three main categories of operations using the DynamoDB API.

1. Control Plane Operations




CONTROL PLANE OPERATIONS

For managing the DynamoDB tables in your account

API Calls:

- ListTable
- DescribeTable
- CreateTable / UpdateTable / DeleteTable

2. Data Plane Operations



DATA PLANE OPERATIONS

Enable you to perform, create, read, update, and delete the following on your DynamoDB tables:

- GetItem API
- BatchGetItem API
- Query
- Scan
- PartiQL

DATA PLANE OPERATIONS

Enable you to perform, create, read, update, and delete the following on your DynamoDB tables:



- PutItem API
- UpdateItem API
- DeleteItem API
- BatchWriteItem API
- PartiQL

3. Transactions Operations

TRANSACTIONS OPERATIONS

For ACID compliance:



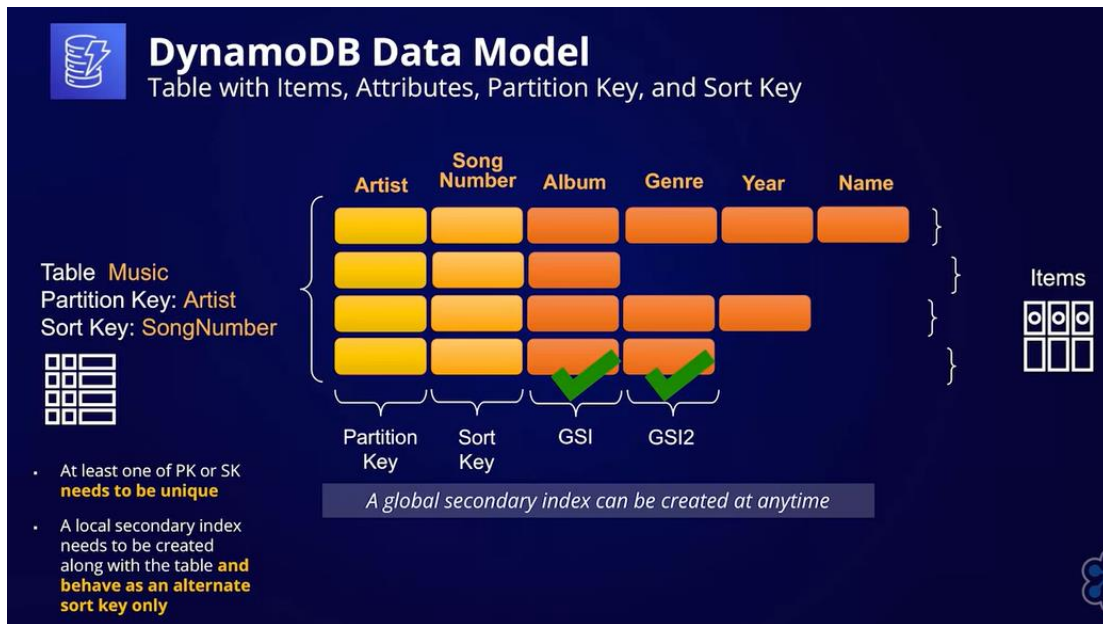
- TransactGetItems API
- TransactWriteItems API
- PartiQL

DynamoDB has two different kinds of secondary indexes, global indexes and local indexes.



Secondary Indexes Compared

| | Global Secondary Index | Local Secondary Index |
|-----------------|--|------------------------------------|
| Definition | Anytime | During table creation only |
| Structure | Allows for alternate partition and sort keys | Allows for alternate sort key only |
| Capacity | Requires its own throughput | Uses the core table throughput |
| Consistency | Eventual consistency only | Allows for strong consistency |
| Quota per Table | 20 | 5 |



DynamoDB segments your data behind the scenes into partitions. When there are a lot of unique values in your table, it's called high cardinality.

Amazon Aurora

Amazon Aurora maintains multiple copies of your data in 3 Availability Zones by default. Amazon Aurora replicates data 6 ways across multiple availability zones. In Amazon Aurora, the master database and the replicas share the same storage layer, so no replication is needed.

Endpoints:

- **Cluster Endpoint:** The cluster endpoint points to the current master database instance. Using the Cluster endpoint allows your application to perform read and write against the master instance.
- **Reader Endpoint:** The reader endpoint load balancers connections across the read replica fleet within the cluster.
- **Custom Endpoint:** A custom endpoint load balancer's connection across a set of cluster instances that you choose and register within the custom endpoint. Custom endpoints can be used to group instances based on instance size or maybe group them on a particular db parameter group.

- **Instance Endpoint:** An instance endpoint maps directly to a cluster instance. Each cluster instance has its own instance endpoint. You can use an instance endpoint when you want fine-grained control over which instance you need to service your requests.



Amazon Aurora:

- AWS's fastest growing service
- Database service with superior MySQL and PostgreSQL engine compliant service
- Separates the compute layer from the storage layer

Amazon Aurora Serverless

Aurora Serverless is ideally suited towards applications which have variable workloads and/or have infrequent data accessing and modification needs.

When provisioning an Aurora Serverless database, you configure lower and upper limits for capacity measured in Aurora Capacity Units (ACUs).

Amazon Aurora Serverless

The underlying compute instances are **automatically started and stopped** based on current demand. Instances can be cold booted in

In terms of high availability - the service is underpinned by the same fault tolerant self healing storage layer. There is **nothing to configure** beyond the capacity settings which if required can be manually tuned

The service will **auto adjust the compute layer** and capacity automatically behind the scenes to ensure that the database is available and has the necessary capacity when required.

If the traffic starts to drop off it will begin scaling down, and if enabled, actually shut down the compute entirely when there's no demand for it.

When the compute is turned off, **you only end up paying for the storage capacity**

Amazon Aurora Serverless

An Aurora Serverless database is configured with a **single connection endpoint** which makes sense - given that it is designed to be serverless - this endpoint is obviously used for all read and writes

An option to consider is the **Web Service Data API feature** - available only on Aurora Serverless databases

The Web Service Data API makes implementing Lambda functions which need to perform data lookups and/or mutations within an Aurora serverless database a breeze

The **AWS CLI has been updated** to allow you to execute queries through it from the command line

Amazon Aurora Serverless

Aurora Serverless performs a **continuous automatic backup** of the database with a default retention period of 1 day - which can be manually increased to a maximum retention period of 35 days.

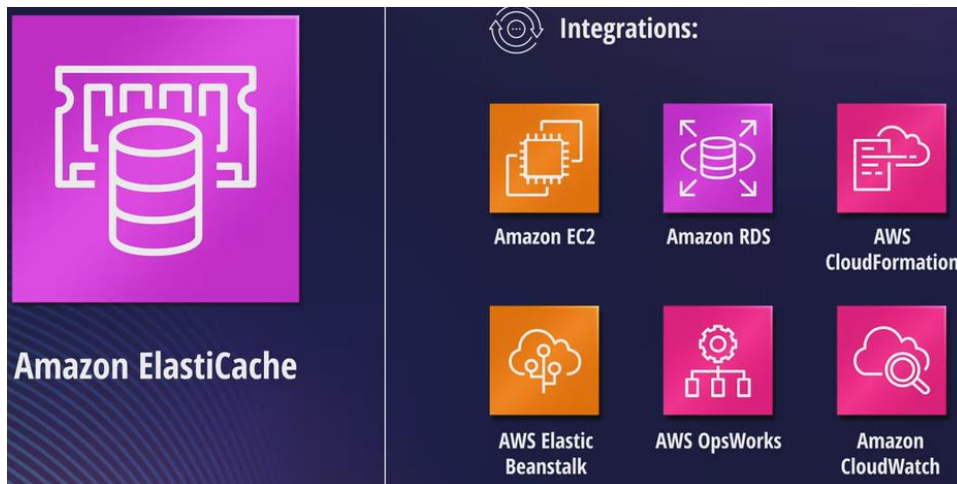
This style of backup gives you the capability of **restoring to a point in time** within the currently configured backup retention period.

Restores are performed to a **new serverless database cluster**

Caching

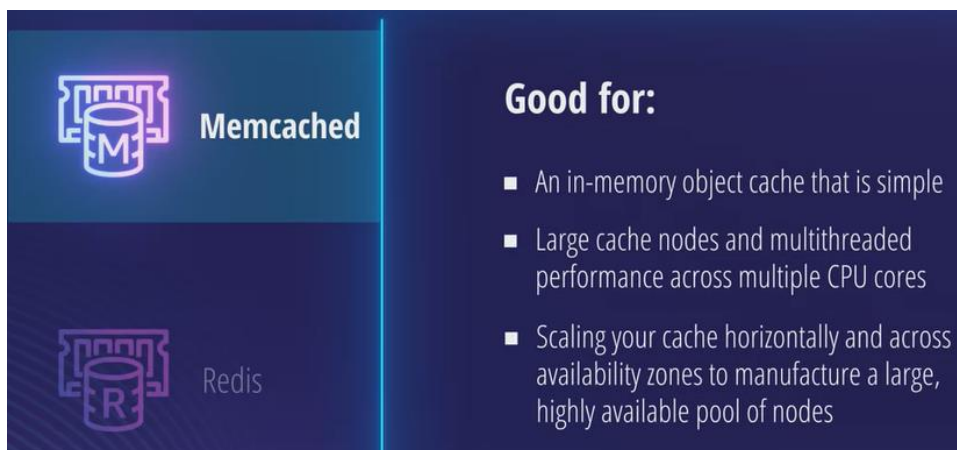
There is in-memory caching and disk caching. In-memory caching is associated with RAM and disk caching is associated with SSDs etc. $\text{Size of Cache} = \text{size of items} * \text{number of items}$.

Amazon ElasticCache



Memcached engine does not support robust authentication or encryption which makes it unsuitable for some compliance standards. M6 and R6 instance types provide the best performance for Memcached implementations.

Redis engine is single threaded and can be configured to automatically detect a failure of the primary node and select the read replica for promotion. For production deployments, Multi-AZ with multiple shards with auto-failover is the best practice.



| | |
|--|---|
|  Memcached | Good for: <ul style="list-style-type: none">■ Complex data types like lists, sets, and bit arrays■ Sorting and ranking in your cached datasets■ Publish and subscribe messaging■ Persistency■ Multi-AZ deployments■ Encryption■ RBAC compliance with PCI DSS, HIPAA, and FedRAMP |
|  Redis | |

Types of Caching:

Lazy Caching and Write Through Caching. Combining both is the best approach.

| | |
|-------------------------------|-----------------------|
| Lazy Caching (Cache Aside) | READ REQUESTS |
| Write Through Caching | WRITE/UPDATE REQUESTS |

Cache Key Expirations:

TTL (time to live).

TTLs are useful for **lazy caching only**

For datasets that change at high speed, **use a TTL measured in seconds** instead of implementing write through caching

Best Practice:

Introduce randomness to the expiration setting of keys

Same TTL = Multiple keys expiring at the same time:

This can cause large sets of cache missed requests and the corresponding request to the primary datastore

- Monitoring cache misses is an important metric to measure the effectiveness of your cache cluster
- **CloudWatch:** Can offer visibility into the behavior of cache

Metrics to measure Cache Success:

Use consistent hashing across cache nodes. Consistent hashing is replication of data to distribute traffic load and handle node failure scenarios.

Amazon ElastiCache for Redis includes CPUUtilization and EngineCPUUtilization metrics in CloudWatch.

Other important metrics reported in CloudWatch are Evictions, Cache misses, Swap Use, Currconnections, BytesUsedForCacheItems

The security metrics reported in CloudWatch for Redis are authentication failures, key authorization failures, command authorization failures.

Amazon MemoryDB for Redis

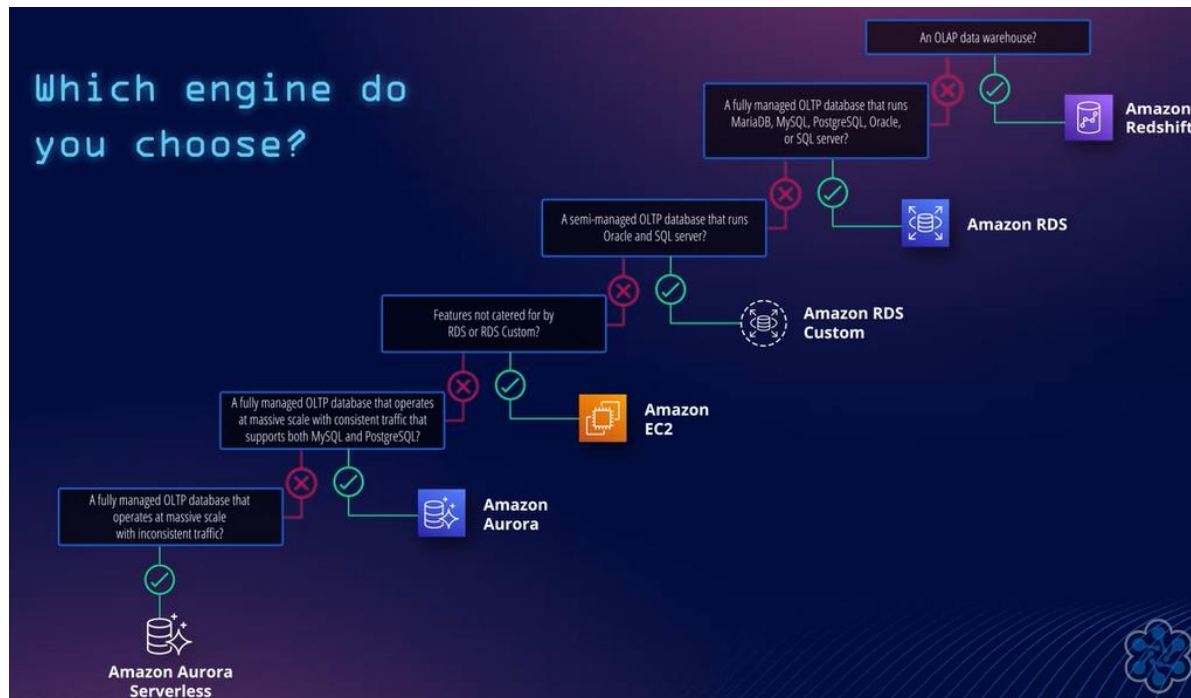
Amazon MemoryDB for Redis



- Fully managed, in-memory, Redis-compatible data store
- Microsecond read and single-digit millisecond write latencies
- Deployed as a cluster serving a dataset that is partitioned into shards
- Each shard has a primary node and up to 5 read replica nodes (can be spread across availability zones for high availability)
- Clusters may have up to 500 nodes, supporting data storage up to 100 TB
- Transaction logs distributed across availability zones to ensure data is durable, consistent, and quickly recoverable
- Data tiering allows you to move less-frequently accessed data to disk
- Supports encryption in transit and at rest
- Snapshots for easy backup and restore of your MemoryDB cluster

Choosing Relational Database on AWS

In Amazon RDS you can't access the underlying operating system or root access. However, in Amazon RDS custom you will have more granular control.



Choosing Non-Relational Database on AWS



Amazon DynamoDB Accelerator (DAX) can be used for caching. Amazon TimeStream is an optimized database that tracks and measures data over a period of time.

AWS – Networking and Content Delivery

CloudFront

CloudFront allows customers to distribute content with low latency and high speed. It is a pay-as-you-use service. When using CloudFront, files are delivered to end-users via a global network of edge locations.

CloudFront works with both static (for ex. Amazon S3) and dynamic (for ex. Amazon EC2) content. CloudFront has three caches in layers: Edge locations, regional edge caches, AWS Origin Shield (it is disabled by default). CloudFront integrates with IAM, Amazon CloudWatch alarms, AWS CloudTrail Logs, and CloudFront real-time IAM standard logs.

CloudFront Patterns:

Pattern 1 – Using CloudFront to cache and secure content when an Application Load Balancer is the Origin.

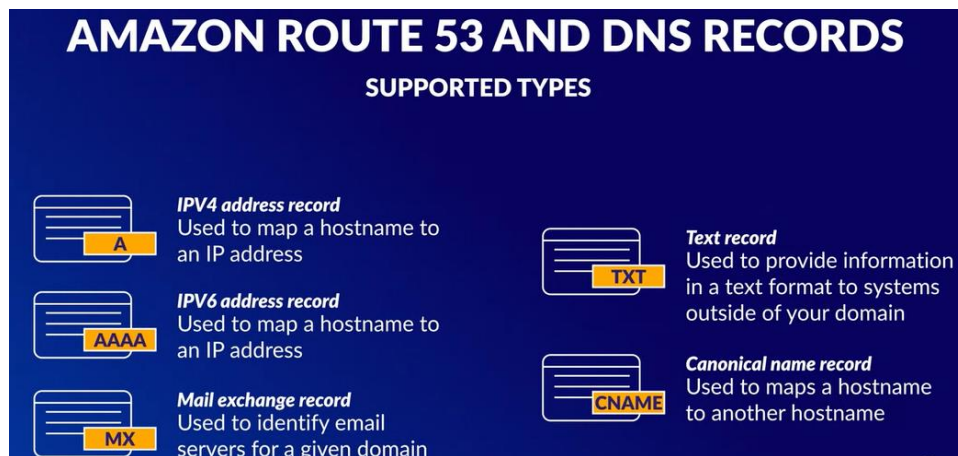
Pattern 2 – Using CloudFront to cache and secure content when an S3 bucket is the origin.

Amazon Route 53

Amazon Route 53 is the domain name management service provided by AWS. The domain name management system (DNS) is responsible for translating domain names to IP addresses. Amazon Route 53 uses edge locations and it's a global service. You don't have to specify a region in configuring Route 53 resources. AWS offers a 100% available SLA for Route 53.

A Public hosted zone defines how traffic is routed on the public internet. A Private hosted zone defines how traffic is routed inside a virtual private cloud (VPC).

Amazon Route 53 creates a set of 4 unique NS records and 1 SOA record in each hosted zone created. The Name Server (NS) records are used to identify the DNS servers for a given hosted zone. The Start of Authority (SOA) record is used to define the authoritative DNS servers for an individual DNS zone. These two records are essential to integrating your domain to the existing DNS system.



One record type that is outside the scope of DNS is the Alias record type.

The Alias record maps a custom hostname in your domain to an AWS Resource. For example, CloudFront distributions, Amazon S3 buckets, and Elastic Load Balancers provide you with a domain name that is internal to AWS. You can use an alias record to define a custom name for that resource.

Amazon Route 53 Health Checks:

If you do not associate a health check with a record, Route 53 treats those records as always healthy. Health checks are checked every 30 seconds unless specified.

Amazon Route 53 Routing Policies:

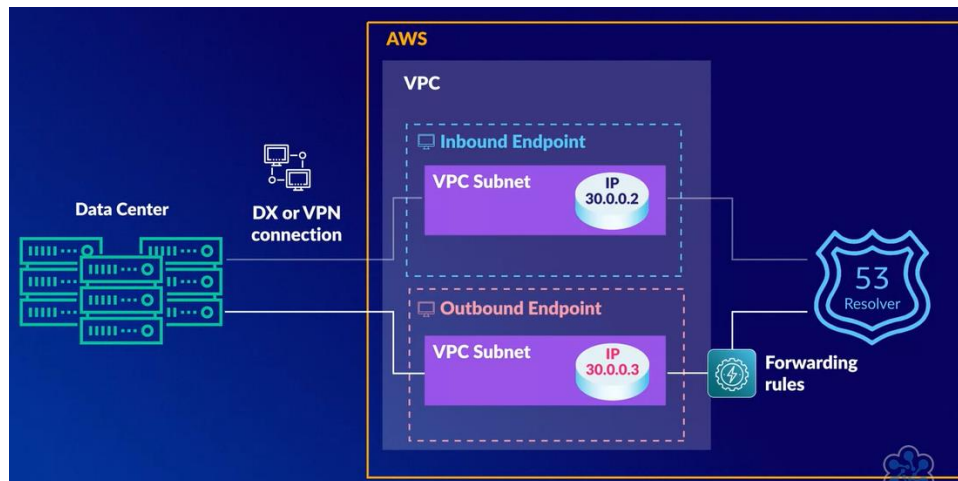
Simple, Weighted, Geolocation, Geoproximity, Failover, Latency, Multivalue Answer.

Amazon Route 53 Traffic Flow:

Traffic flow simplifies the process of creating and maintaining records in large and complex configurations. Geo-proximity routing policy is available only if you use traffic flow.

Amazon Route 53 Resolver:

The Route 53 resolver is the DNS service for VPCs that integrates with your data center.



The Route 53 Resolver DNS firewall is a managed firewall service that inspects and filters traffic coming from your VPC.

Route 53 Application Recovery Controller:

Route 53 application recovery controller continuously monitors an application's ability to recover from failures and controls application recovery across multiple availability zones, regions, and possibly your own data center environments. This can be done using readiness checks, routing controls, and safety rules.

Amazon API Gateway

It's serverless and fully managed, mainly used for building, publishing, monitoring, securing, and maintaining APIs.

Amazon API Gateway has three different endpoint types: Edge-Optimized API Endpoint (all requests will be routed to the closest CloudFront bringing your API closer to your customer), Regional API Endpoint (works well when you want your API calls to come from a single AWS Region), Private API Endpoint (can only be accessed from within your VPC).

There are two different ways you can connect with your backend: proxy integration and direct integration. A proxy integration is basically just a pass-through setup, where API Gateway will pass the request straight to the backend without trying to modify anything along the way. The direct integration allows you to make changes to the request as it passes through API Gateway and to modify the response on its way back to the client.

Amazon says:

"HTTP APIs are ideal for:

1. Building proxy APIs for AWS Lambda or any HTTP endpoint
2. Building modern APIs that are equipped with OIDC and OAuth 2 authorization
3. Workloads that are likely to grow very large
APIs for latency-sensitive workloads

REST APIs are ideal for:

Customers looking to pay a single price point for an all-inclusive set of features needed to build, manage, and publish their APIs."

Amazon API Gateway has several authorization options: IAM Authorizer, Lambda Authorizer, Cognito Authorizer, JWT Authorizer.

For Security, API Gateway is integrated with AWS WAF (Web Application Firewall).

AWS – Analytics

Amazon Kinesis

Kinesis makes it easy to collect, process, and analyze various types of data streams such as event logs, social media feeds, clickstream data, application data, and IoT sensor data in real time or near real-time.

Access to Kinesis is controlled using AWS Identity and Access Management.

Using the AWS Key Management Service (KMS) data is encrypted and decrypted in the stream and stored in Amazon S3 or Redshift.

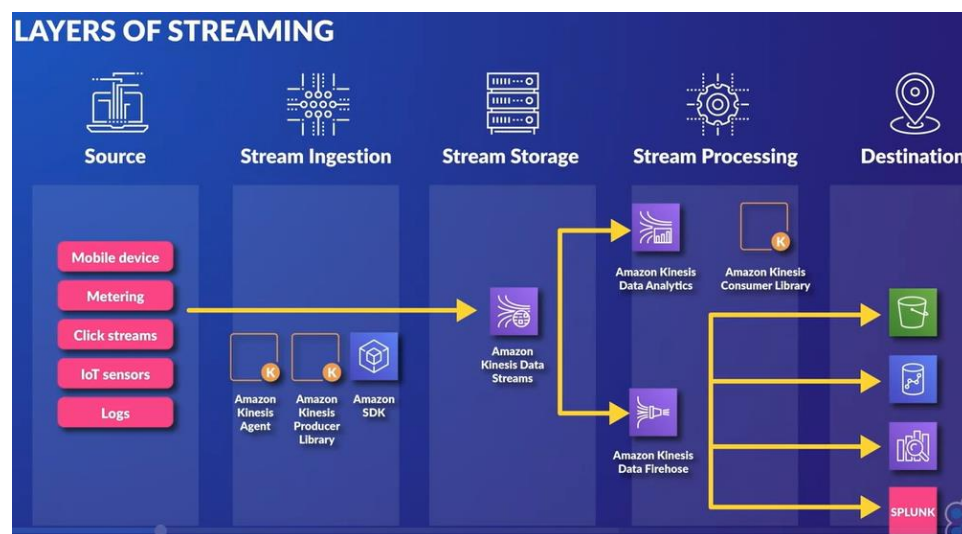
Data in transit is protected using TLS, the Transport Layer Security Protocol.

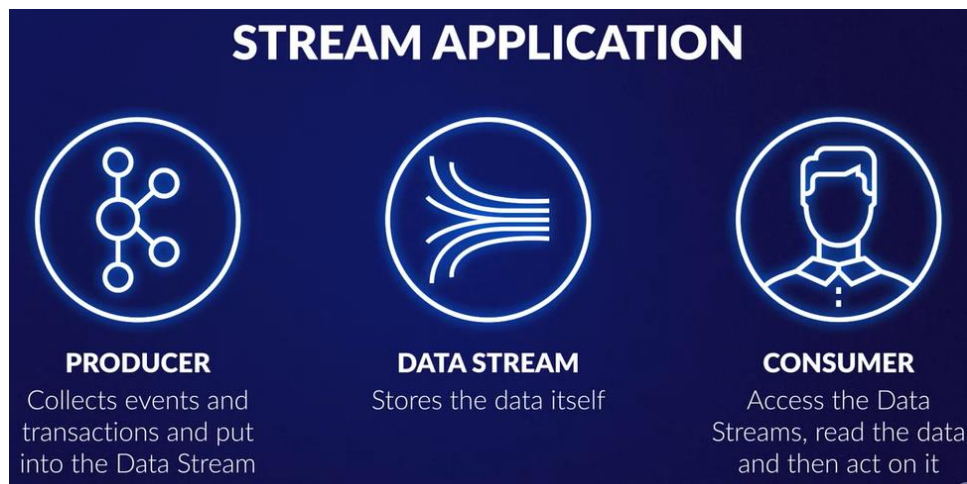
Amazon Kinesis is composed of four services, Kinesis Video Streams, Kinesis Data Streams, Kinesis Data Firehose, and Kinesis Data Analytics.

Kinesis Video Streams is used to do stream processing on binary-encoded data, such as audio and video.

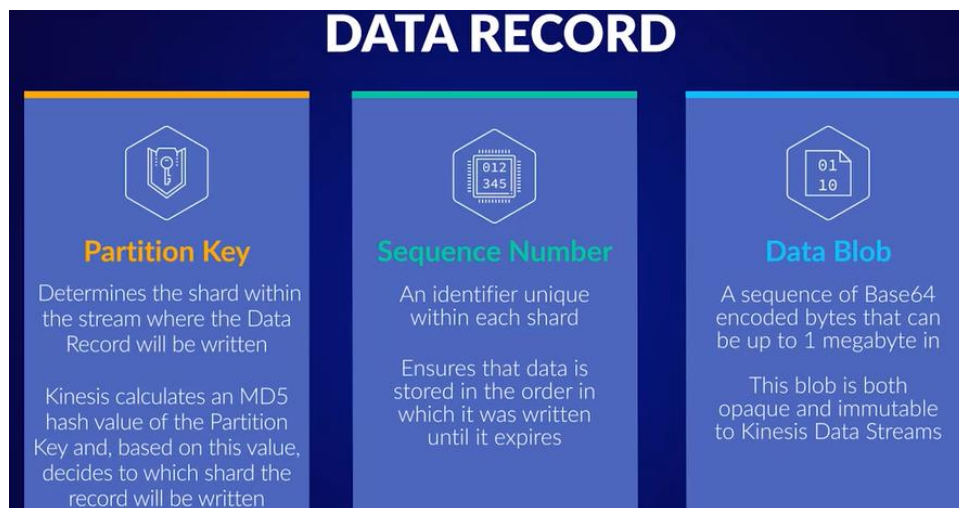
Kinesis Data Streams, Kinesis Data Firehose, and Kinesis Data Analytics (helps you write SQL queries on incoming data streams) are used to stream base64 text-encoded data, such as logs, click-stream data, social media feeds, financial transactions, in-game player activity, geospatial services, and telemetry from IoT devices.

There are five layers of streaming data: Source, Stream Ingestion, Stream Storage (Data Records are immutable inside Amazon Kinesis Data Streams), Stream Processing, and the Destination.





Stream processing is used to collect, process, and query data in either real time or near real time to detect anomalies, generate awareness, or gain insight.



A shard is a collection of data records in a stream, and each shard can have up to 1,000 records. For writes per shard, the limit is 1,000 records per second up to a maximum of 1 megabyte per second. For reading, a shard can sustain the rate of up to a maximum of two megabytes per second.

If more data records need to be added, it is possible to add more shards to a stream. This process is called resharding.

Streams can be created using the AWS Console, SDK, AWS CLI. The stream must have a unique name and at least one shard. Each shard in a stream has a Shard ID, a Hash Key range, and a starting Sequence Number.

HTTP/2 is a binary protocol that enables new features designed to decrease latency and increase throughput.

Producers, when they put records into a stream, have a pair of limits. Data can be written at a rate of 1 megabyte per second per shard or there can be 1,000 writes per shard per second.



CONSUMERS: LIMITS

STANDARD CONSUMER

Uses a polling method to get data from a stream

Each shard can:

- Support up to 5 API calls per second
- Return up to 2 megabytes per second, to a total of 10,000 records per second



CONSUMERS: LIMITS

ENHANCED FAN-OUT

Uses a push mechanism to send data to Consumers

There can be up to 20 Consumer Applications registered to a stream

Each Consumer gets 2 megabytes per second per shard of throughput

This throughput comes at a cost

Kinesis Data Streams:

Amazon Kinesis Data Streams enables you to build custom applications that process or analyze streaming data for specialized needs. It provides real-time data streams that can handle high volumes of data. Kinesis Data Streams throughput is based on the number of shards it has. In Kinesis Data Streams, shard capacity and auto scaling are both manual processes that require configuration changes.

Hot shards have many reads and writes per second, while cold shards have a low number of reads and writes per second. A hot shard can cause throttling. Cold shards waste money.

Adding a shard is called Shard Splitting. Removing a shard is Shard Merging.

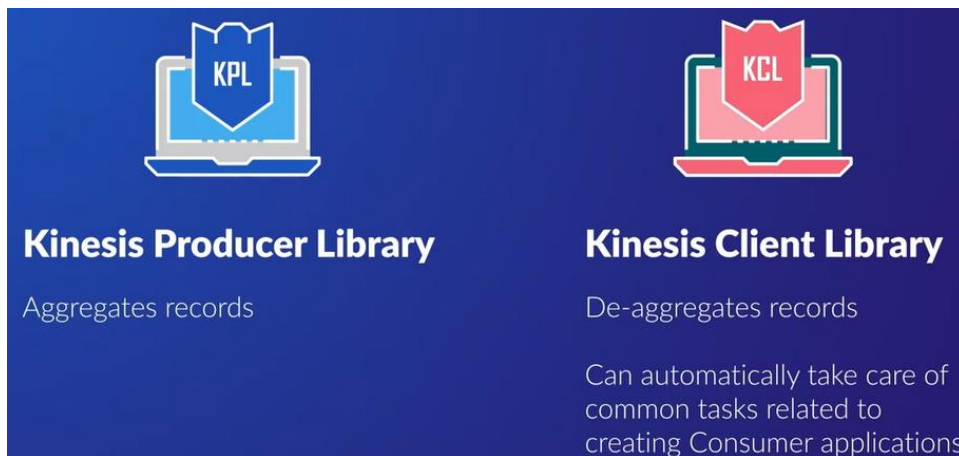
Shard splitting can be done on a stream that is active. However, only one split or merge operation can happen at a time.

When a shard is split or merged there are Parent Shards and Child Shards. The Parent Shards are closed for writing but are available for reading until they expire.

When a Parent Shard is closed, write operations intended for it are rerouted to the Child Shard.

Shards can be in one of three states, Open, Closed, and Expired.

The KPL (Kinesis Producer Library) is designed to work with Kinesis Data Streams, not Kinesis Data Firehose.



Use the Principle of Least Privilege when provisioning resources inside AWS.

Kinesis Data Firehose:

Kinesis Data Firehose enables you to load streaming data into Amazon S3, Amazon Redshift, and Amazon Elasticsearch services. Kinesis Data Firehose does not need shards.

Amazon OpenSearch Service:

Amazon OpenSearch Service is the successor to the Amazon Elasticsearch Service.

Amazon OpenSearch Service



- Fully-managed service for deploying OpenSearch clusters
- Open-source search, analytics, and visualization suite created by Amazon that is based on Elasticsearch and Kibana
- Index and search large collections of data for search, monitoring, and analytics
- Provides Elasticsearch and OpenSearch APIs
- Integrates with visualization tools like Kibana and OpenSearch Dashboards
- Cluster deployments across 2 or 3 availability zones offer high availability
- Ingest streaming data from CloudWatch Logs, Kinesis Data Firehose, and other AWS services, including S3 and DynamoDB
- Cost-effective tiered storage for read-only archive data
- Serverless option requires no infrastructure



Amazon Athena:

Athena supports encryption methods such as Server-side encryption with an S3-managed key (SSE-S3), Server-side encryption with a KMS key (SSE-KMS), Client-side encryption with a KMS key (CSE-KMS).

Amazon Athena



- Enables you to interactively query data in S3 and other data sources using standard Structured Query Language (SQL) syntax
- Allows you to analyze petabytes of data without the need to provision any infrastructure
- Utilizes DDL to define tables and supports querying CSV, JSON, Parquet
- Partition keys allow you to query large datasets by any column
- Integrates with EMR, Glue, QuickSight, and Redshift Spectrum
- Supports querying compressed data stored in S3
- Can use SDKs, Athena API, or ODBC/JDBC driver to interact with Athena

AWS – Management and Governance

Amazon CloudWatch

Amazon CloudWatch provides information about the health and operational performance of your applications and infrastructure. It's heavily used by those in an operational role and site reliability engineers.

The Unified CloudWatch Agent allows the collection of logs from EC2 instances and on-premises servers too.

CloudWatch Dashboards:

Using the AWS Management console, the AWS CLI, or the PutDashboard API, you can build and customize a CloudWatch Dashboard using different visual widgets displaying metrics and alarms relating to your resources.

CloudWatch Metrics and Anomaly Detection:

CloudWatch Metrics enables you to monitor a specific element of an application or resource over a period while tracking some data points. Custom metrics are regional.

Anomaly detection allows CloudWatch to implement machine learning algorithms against metric data to help detect any activity that sits outside of the normal baseline parameters.

Alarms can be created based on normal pattern and triggered when they are “Outside the band”, “Greater than the band” or “Lower than the band”.

CloudWatch Alarms:

They allow you to implement automatic actions based on specific thresholds.



CloudWatch EventBridge:

Connects your own applications to a variety of different targets, typically AWS services, to allow you to implement a level of real-time monitoring.

An event is anything that causes a change to your environment or application.

Elements of CloudWatch EventBridge are Rules, Targets, and Event Buses.

Rule - The rule acts as a filter for incoming traffic and routes traffic to multiple targets, however the target must be in the same region.

Target – Targets are nothing but AWS Lambda, SQS, Kinesis or SNS.

Event Bus - It is the component that receives the Event from your applications and your rules are associated with a specific event bus.

CloudWatch Subscriptions:

It provides access to a real-time feed of log events from CloudWatch logs. It can deliver those logs to other services such as Kinesis streams, Firehose, or Lambda for custom processing and analysis.

CloudWatch Logs:

CloudWatch Logs gives you a centralized location to house all your logs from different AWS services.

CloudWatch Insights:

Insights - They provide the ability to get more information from CloudWatch data.

Log Insights - This is a feature that can analyze your CloudWatch logs at scale in seconds using interactive queries delivering visualizations.

Container Insights - Container Insights allow you to collate and group different metric data from different container services and applications within AWS, for example, the Amazon Elastic Kubernetes Service, (EKS) and the Elastic Container Service (ECS).

Lambda Insights - This feature provides you the opportunity to gain a deeper understanding of your applications using AWS Lambda.



Amazon CloudWatch

- Analyze, identify, monitor and report on the data captured from within your CloudTrail log files
- Offers the capability to gather additional intelligence and insight from your logs allowing you to proactively detect security risks, identify performance problems, and resolve potential issues

AWS CloudTrail

AWS CloudTrail logs account activity. CloudTrail assumes the IAM role you specify to deliver API activity to CloudWatch Logs. CloudTrail is helpful in troubleshooting operational and security incidents. It delivers logs to S3 bucket every 5 minutes. There are 3 types of events that CloudTrail categorizes and tracks:

Management Events – Similar to Control Plane Operations.

Data Events – Similar to Data Plane Operations.

CloudTrail Insight Events – Detects unusual activity.

There are 3 different types of Trails that you can create:

All Region Trail - It's a trail that will apply to all regions.

Single Region Trail - It's a trail that will apply to a single region.

AWS Organization Trail - Captures all events from all accounts that belong to the AWS Organization. It can be configured to capture events from a single region, or all regions.

Another component of AWS CloudTrail is CloudTrail Lake. CloudTrail lakes allow you to collectively store, review and query different events that have been generated by your resources, and then accumulate them into event data stores, keeping them for up to 7 years.

Benefits of CloudTrail are: Improving security, consolidate records, enhanced visibility, auditing and compliance.

There are 3 AWS managed policies related to AWS CloudTrail:

CloudTrailServiceRolePolicy - This service-linked role performs specific operations on your behalf. It's not possible to attach this policy to any users, groups or roles.

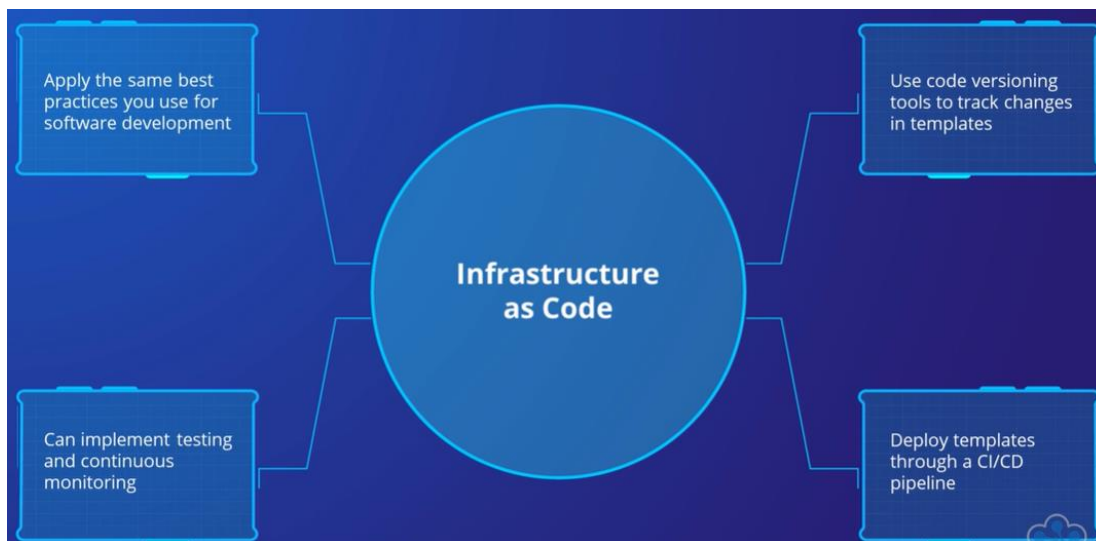
AWSCloudTrail_FullAccess - Provides all the necessary actions allowing you to configure AWS CloudTrail.

AWSCloudTrail_ReadOnlyAccess – Only four actions are allowed cloudtrail:Get, cloudtrail:describe, cloudtrail:List, cloudtrail:LookupEvent.

AWS CloudFormation

You can automate the creation, the updating, and the deletion of your infrastructure and its configurations all in one place. It does not support all AWS services.

If an AWS CloudFormation template fails to create a resource, CloudFormation reverts the stack to the last known stable configuration.



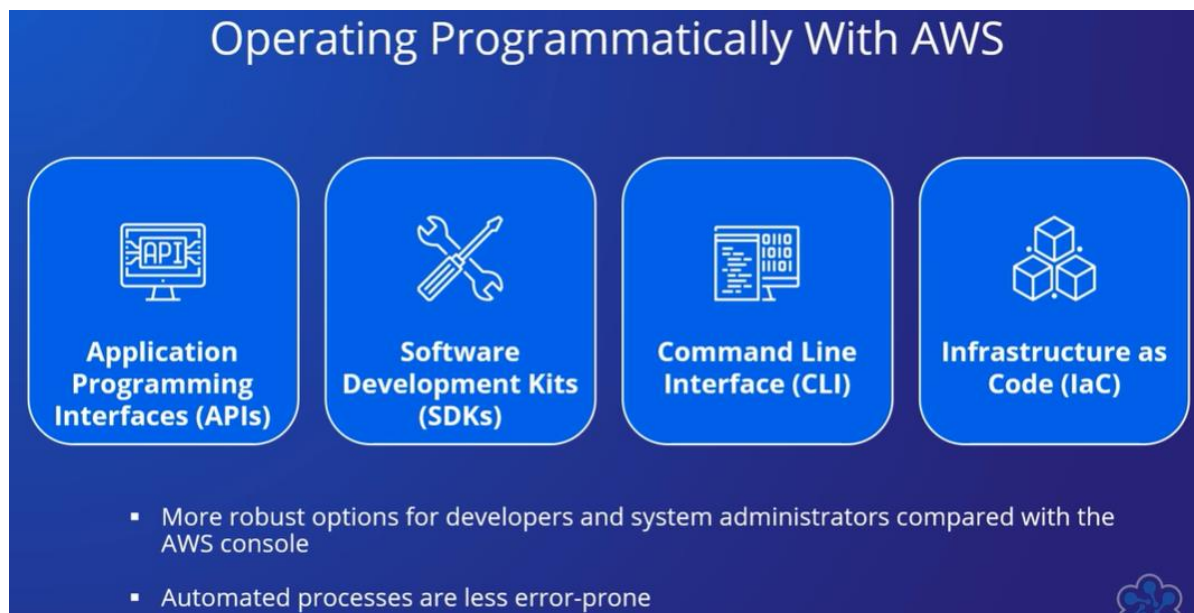
CloudFormation Template:

Templates are nothing but stacks in CloudFormation. The valid sections of a CloudFormation template are: AWSTemplateFormatVersion, Description, Parameters, Mappings, Conditions, Transform, Resources, Outputs.

VPC Flow Logs

VPC Flow Logs captures IP traffic information that flows between your network interfaces of your resources within your VPC. The logs are stored within CloudWatch logs.

Operating Programmatically with AWS



AWS CLI

AWS CLI provides more flexibility to access and manage AWS resources.

JSON and YAML formats are best with programming languages. Text is best for scripting languages, table is best for humans looking to read data, and YAML-stream is best for large data sets, as it provides a faster and more responsive output.

Software Development Kit (SDK) vs Cloud Development Kit (CDK)

CDK is an infrastructure-as-code tool that allows developers to define AWS resources. For Ex. Cloudformation

SDK is a set of tools for developers to interact with AWS services, such as creating API calls and Lambda functions.

AWS Systems Manager

You can use automation to manage traditional and cloud workloads by performing essential setup, maintenance and management tasks, while maintaining complete visibility and control over your entire machine farm independent of operating system, location and number of instances. Most of its functionality is free.

The Systems Manager Agent is the software required to be installed and configured on all instances for them to be called managed instances.


A managed instance is an instance with the ability to communicate and be operated by Systems Manager.

You can find all the details of managed instances in the fleet manager. You can also start a session for a managed instance from the session manager.

The Systems Manager Run Command will permit you to execute a command on one or more of your managed instances.

Using systems manager parameter store you create a new parameter with a parameter type and specify the corresponding value (Ex. `{{ssm: parameter-name}}`). All these parameters are stored centrally. You can reference parameters in your commands or application code using API calls.

In AWS Systems Manager Patch Manager, patches are defined using patch baselines.



PATCH MANAGER

- Allows automation for patching managed instances with both security and reliability updates
- Use to apply updates to both operating system and applications running on managed instances
- Patch fleets of managed instances by operating system type



STATE MANAGER

- Allows you to control how configurations are applied
- Used to enforce enterprise-wide compliance by ensuring a desired state is continuously applied to your managed instances



STATE MANAGER

- Network settings or bootstrap instances can be configured with software modules at startup
- Because it reapplies configuration states, it can maintain configuration consistency



Secrets Manager



Holds, protects and contains sensitive information for you



Allows other services and applications to call for the secret via a simple API call



No need to hard-code any secret or credentials within your applications

Secrets Manager is fully integrated with AWS's Identity and Access Management (IAM).

Encryption is enabled by default in secrets manager whereas encryption is not enabled by default in parameter store. Encryption is carried out by AWS Key Management Service (KMS).

Automated rotation of keys is enabled by default in secrets manager, whereas keys must manually rotate in parameter store.

From a deployment use case, whichever service you are using Parameter Store, or Secrets Manager, the values are referenceable using AWS CloudFormation.

Parameter store secret size is 4kb (free) and 8kb (chargeable). Secrets Manager secret size is 10kb (chargeable).

AWS AppConfig

AWS AppConfig is a feature of AWS Systems Manager.

AWS AppConfig



- Controls when and how to deploy feature flags and app updates
- Ensures your applications are always running in a desired state and that changes are made in a controlled, predictable manner
- Organizes and manages a set of related configuration data for your application that can then be applied to one or more environments (dev, staging, etc.)
- Integrates with CloudWatch Alarms to roll back configuration updates if an alarm is triggered during a deployment
- Configuration profiles can include feature flags, connection strings, or other parameters and allow you to apply updates to a set (or percentage) of instances before rolling them out to your entire fleet
- Validates configuration format and values before deployment
- Can track and roll back configuration changes for quick recovery

AWS Cloud Development Kit (CDK)

AWS Cloud Development Kit (CDK)



- Open-source framework that allows you to define AWS resources and infrastructure in code
- Generates equivalent AWS CloudFormation templates at compile time
- Enjoy all the benefits of CloudFormation without maintaining YAML templates
- Supports familiar programming languages (Python, C#, Java, etc.) allowing developers to use existing constructs like functions, loops, and conditionals
- Provides a set of libraries known as constructs to define AWS resources like S3 buckets, DynamoDB tables, Lambda functions, and API Gateway APIs
- Developers can create and customize constructs for reuse and sharing
- Also provides a CLI that can be used for common tasks like creating new CDK applications and deploying them to specific environments

AWS – Developer Tools

Continuous Integration

If developers are continuously integrating their code to GIT, then it's called continuous integration (CI).

Version control or source control or revision control are the same.

Source code analysis tools can check for known security holes and identify the use of vulnerable dependencies.

Continuous Delivery or Continuous Deployment

Continuous delivery (CD) is a way of building software such that it can be deployed to a specified environment (for ex. dev, stage, prod). Tools are jenkins, travis CI etc.

Configuration management manages your servers and infrastructure in a scriptable way. Tools are Ansible, chef, puppet, docker etc.

Jenkins is generally better for testing software projects, while Ansible is better for automating the deployment, configuration, and management of infrastructure and applications. Jenkins and Ansible can be used together as part of a larger testing and deployment pipeline. For example, in a Jenkins CI/CD pipeline, Ansible can provision infrastructure and deploy apps, while Jenkins orchestrates the entire process.

In canary deployments, you deploy the latest version of your software into a production environment, and you have a router select a group of users to route to it. You can see how it behaves before making the decision to roll it out further or remove it entirely.

Mutable vs Immutable Servers

Mutable servers can be changed after deployment, and immutable servers cannot. Mutable servers scale less effectively than immutable servers.

Mutable servers offer low overhead, and the ability to rollback changes if version control is maintained. Ad-hoc commands are relatively simple with mutable servers, and with the wide variety of configuration management tools available, mutable servers can be relatively easy to manage.


Immutable servers more often offer the ability to work with cloud scaling features, while mutable servers most often do not. With immutable servers, no additional testing is required for OS level changes, but this is required with mutable servers.

AWS Amplify

AWS Amplify is a mobile and web toolset that AWS offers to help developers create scalable front-end applications.

AWS Amplify

- Provides a set of tools and services for full-stack application development on AWS
- Allows you to configure, deploy, and host applications that use AWS services and run on iOS, Android, React Native, or the web
- Includes an open-source framework with libraries, UI components, and a CLI
- Supports authentication, analytics, offline data, and push notifications
- Amplify Studio offers a visual design experience and works with the CLI
- Includes a hosting service with support for full-stack CI/CD
- Supports libraries and frameworks, including React, Angular, and Vue
- Figma integration allows importing design prototypes into React code
- Integrates with AWS Device Farm for testing on real-world devices




Cloud9

Cloud9 is an IDE which can be accessed using an internet browser.

AWS CloudShell

AWS CloudShell




- Browser-based Linux shell experience that provides secure, pre-authenticated access to AWS CLIs and other tools
- Makes it easy to interact with AWS services without needing to set up a local development environment first
- Security is pre-configured based on the credentials of the user currently logged in to the AWS console
- Access in any supported region by clicking the  icon in the top navigation
- Pre-configured with the AWS CLI and CLIs for Elastic Beanstalk, ECS, and SAM
- Can access Python and Node.js and install additional tools that are not included by default
- Home directory includes 1 GB of persistent storage

AWS CodeCommit

Fully Managed and serverless. It is like GIT, and you should have IAM permission to access it. AWS CodeCommit integrates with Amazon EventBridge, AWS CodePipeline.

When an event happens in an AWS CodeCommit environment, you can choose to do two things:co You can be notified of the event, or you can take action.



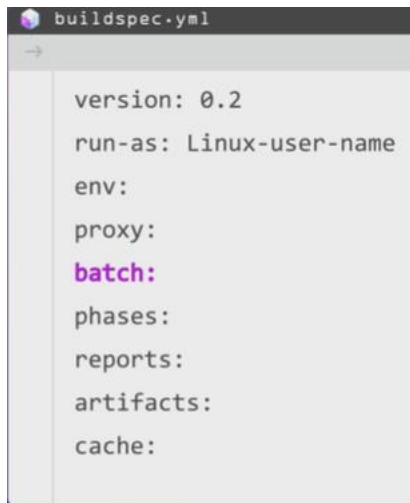
REPOSITORY TRIGGERS

- Based on repository events, take actions such as SNS notifications or Lambda functions when:
 - ☐ Creating or deleting a branch or tag
 - ☐ Pushing to an existing branch, or
 - ☐ Actioning any repository event
- Core differences between repository triggers and notifications are that triggers:
 - ☐ Can easily trigger a Lambda function
 - ☐ Don't use CloudWatch Events rules to evaluate repository events
 - ☐ Are limited in what they can do and generally trigger based on branch or tag events

AWS CodeBuild

AWS CodeBuild is a fully managed build service. It's designed to compile your source code, run unit tests, and then create an artifact that can eventually be deployed. There are three core concepts of CodeBuild: a CodeBuild project, a build environment (here you can select a docker image or an image from ECR), and a buildspec file. When a build project is completed, you have the option to save your build artifacts to an S3 bucket. A build badge in AWS CodeBuild displays the status of the latest build for a project.

In the yaml file, under phases section there is install, pre-build, build, and post-build.



AWS CodeDeploy

This service is used for rolling out changes effortlessly and quickly. There are three key terms in CodeDeploy: Application, Deployment Group, and Deployment. A code deploy agent is mandatory for deployments on EC2/On-Premises while Lambda and ECS do not need code deploy agent. It integrates with CloudWatch and SNS. Appspec files for EC2/On-Premises contains five sections: version, os, files, permissions, and hooks. Appspec files for Lambda and ECS contain three sections: version, resources, and hooks.

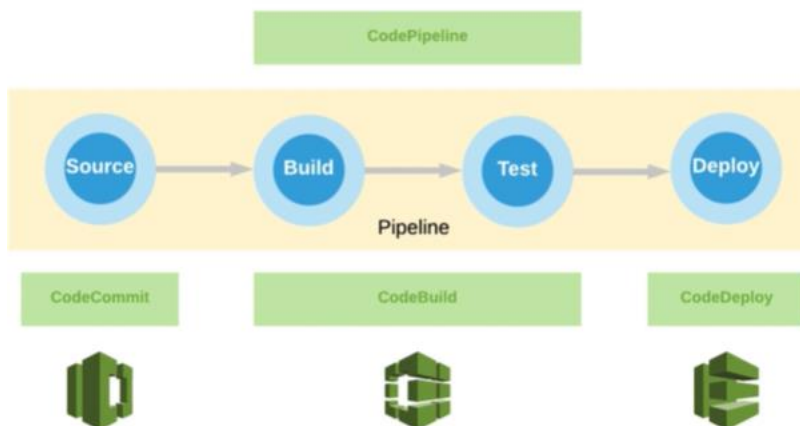
Select blue-green or in-place deployment when configuring a new deployment type in AWS CodeDeploy. An in-place deployment is a deployment strategy that updates the application version without replacing any infrastructure components. In blue-green deployment, the new version is released alongside the old version, and then the traffic is switched to the new version.

A deployment configuration is a constraint that determines how a deployment progresses through the instances in a deployment group.



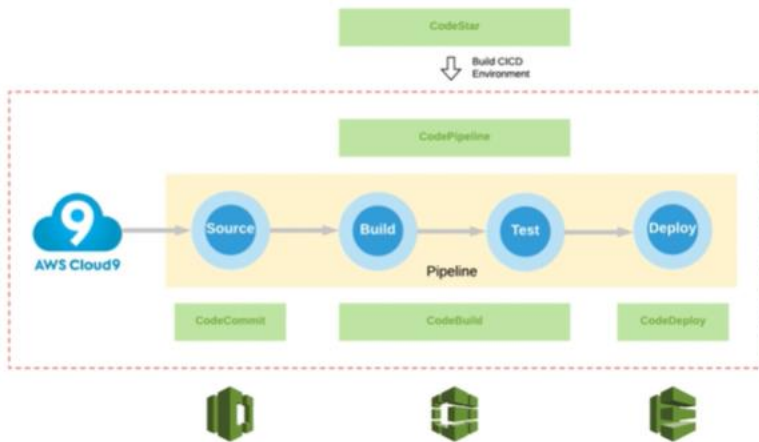
AWS CodePipeline

AWS CodePipeline is a fully managed, continuous delivery system used to automate the building, testing, and deployment of your code. CodePipeline has six action types: approval, source, build, test, deploy, and invoke. It integrates with CloudWatch. ECR and Gitlab are valid sources for AWS CodePipeline.



AWS CodeStar

It has prebuilt CI/CD workflow templates (for ex. Python application) that will launch your CI/CD workflow in minutes.



AWS X-Ray

AWS X-Ray is a valuable tool for developers who need to debug and optimize the performance of microservices in distributed architecture. X-Ray provides an end-to-end view of requests known as a service map that illustrates how the different parts of your application interact. Within the service map, the health of each node or microservice is represented with colors. Green equals successful calls, red equals server faults, yellow equals client errors, and purple represents throttling errors. AWS Lambda can be integrated with X-Ray for built-in distributed request tracing.

AWS X-Ray Instrumentation Concepts

| | |
|--------------------|---|
| Segments | Portions of the trace that correspond to a single service |
| Subsegments | Remote call or local compute sections within a service |
| Traces | End-to-end path of a request through your application |
| Filters | Use filter expressions to view a service map or traces for requests that have performance issues or that relate to specific requests. |
| Annotations | Business data added to trace - indexed for filtering |
| Metadata | Business data added to trace - but not indexed for filtering |
| Exceptions | Application error message and/or stacktrace |