## EC2

**Auto Scaling**

* You create collections of EC2 instances, called Auto Scaling groups.
* 
     An illustration of a basic Auto Scaling group.
    
* Min size – Number of instances minimum scaled in
* Max size – Number of instances maximum scaled out
* Desired Capacity – Always ASG ensures to have the specified number of instances
* ASG can be configured to use Multi AZ
* An Auto Scaling group can contain EC2 instances in one or more Availability Zones within the same Region. However, **Auto Scaling groups cannot span multiple Regions.**

**Benefits**

* Fault tolerance: Detect unhealthy instances, terminates it and re-launch /replace it as per configurations (AMI etc.,)
* Availability: Always ensures to have right amount of EC2 instances to handle the traffic
* Cost management: Dynamically scale-in/out EC2 instances based on demand. So the pay is reduced.

**Use cases:**

Example: Covering Variable Demand


     An example showing how Amazon EC2 Auto Scaling can adjust capacity as needed.
    

Example: Web App Architecture


     An illustration of a basic Auto Scaling group.
    

Example: Distributing Instances Across Availability Zones

Key components in auto scaling:

* Auto Scaling group
* Configuration templates
* Scaling options

**ASG:**

* The size of an Auto Scaling group depends on the number of instances you set as the desired capacity.
* Health checks/EC2 status checks
* ASG Perform periodic health checks by /Health. It can be **either instance status/system status checks.**
* Health check options can be suspended for ASG
* System status check fails if
  + Loss of network connectivity
  + Loss of system power
  + Software issues on the physical host
  + Hardware issues on the physical host that impact network reachability
* Instance status check fails if
  + Failed system status checks
  + Incorrect networking or startup configuration
  + Exhausted memory
  + Corrupted file system
  + Incompatible kernel
* CLI to view health check status
  + **aws ec2 describe-instance-status**
  + aws ec2 describe-instance-status --filters Name=instance-status.status,Values=impaired
  + **aws ec2 describe-instance-status --instance-ids i-1234567890abcdef0**
* EC2 Query API
  + **DescribeInstanceStatus**
* Unhealthy status (other than “running”)
  + Impaired
  + Stopping
  + Stopped
  + Terminating
  + Terminated

**Launch Configurations**

* Included are the ID of the Amazon Machine Image (AMI), the instance type, a key pair, security groups, and the other parameters that you use to launch EC2 instances.
* **AWS recommend that you use launch templates** instead of launch configurations to ensure that you can use the latest features of Amazon EC2, such as T2 Unlimited instances.

**Scaling Options**

* Maintain current instance levels all the time
* Manual scaling, ie, min, max & desired capacity
* Scale based on a schedule
* Scale based on demand

Detailed Monitoring of EC2

* Basic – **5 minutes interval without cost**
* Detailed – **1-minute interval with cost**

Instance Metadata Retrieval

* <http://169.254.169.254/latest/meta-data/>
* For example, you can acces**s the local IP address of your instance from instance metadata to** manage a connection to an external application.

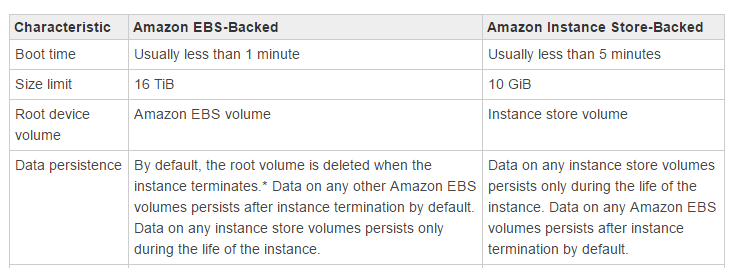
Instance User data Retrieval

* Limited to **16 kb**
* [**http://169.254.169.254/latest/user-data**](http://169.254.169.254/latest/user-data)

Instance Dynamic data Retrieval

* <http://169.254.169.254/latest/dynamic/>

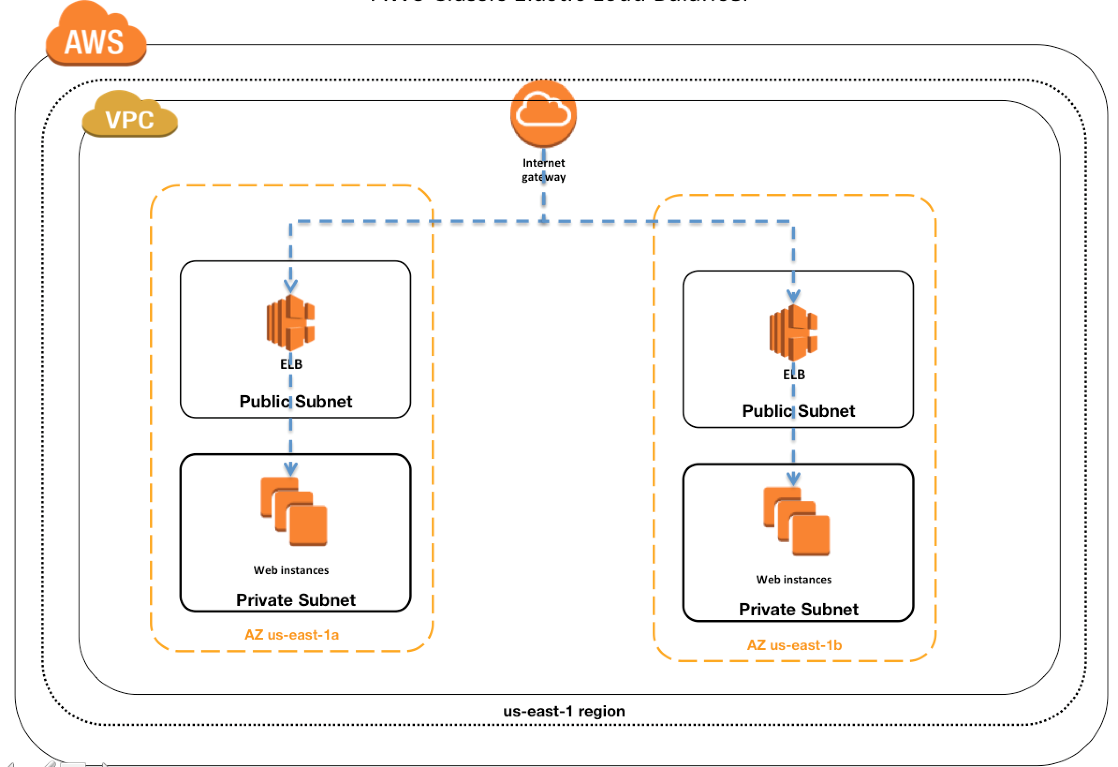
### EBS Backed vs Instance Store



AMIs

* Can be shared to specific list of accounts
* Can be a public AMI

ELB association



## Elastic IPs

* Static IPV4 address associated to our AWS account
* With an Elastic IP address, **you can mask the failure of an instance or software by rapidly remapping the address** to another instance in your account.
* It’s a public IPV4 address reachable from internet
* No support for IPV6 EIPs

**Basics**

1. Needs to be associated with network interface
2. If it’s associated with EC2 and running, for 1 EIP, Amazon wont charge. But additional EIPs will be charged
3. If we grabbed it from pool and not released without using it, then hourly charges
4. Region specific
5. Maximum 5 per region

### CLI Commands

Allocate

allocate-address

Describe

describe-addresses

Tags

create-tags

Associate

associate-address

Disassociate

disassociate-address

Release

release-address

### Recovering an Elastic IP Address

**If you have released your Elastic IP address, you might be able to recover it**. The following rules apply:

* You cannot recover an Elastic IP address if it has been allocated to another AWS account, or if it will result in your exceeding your Elastic IP address limit.
* You cannot recover tags associated with an Elastic IP address.
* You can recover an Elastic IP address using the Amazon EC2 API or a command line tool only.

## S3 – Simple Storage Service

* Amazon S3 is designed for **99.999999999% (11 9's**) of durability,
* Maximum **of 5 terabytes**
* The largest object that can be uploaded in a single PUT is 5 gigabytes. For objects larger than 100 megabytes, use multipart upload.
* S3 **Inventory report that lists your stored objects within an S3 bucket or with a specific prefix**, and their respective metadata and encryption status.
* S3 Batch Operations makes it simple, whether you store thousands of objects or a billion, to manage your data in Amazon S3 at any scale.
* With S3 Versioning, you can easily preserve, retrieve, and restore every version of an object stored in Amazon S3, which allows you to recover from unintended user actions and application failures.
* With **S3 Cross-Region Replication (CRR), you can replicate objects (and their respective metadata and object tags) into other AWS Regions** for reduced latency, compliance, security, disaster recovery, and other use cases.
* **WORM** policy – **Write Once Read Many**
* When bucket is re-created in us-east-1, it resets ACLs
* Maximum **100** buckets
* Key
  + A key is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. "**bucket + key + version**"
* After you have created a bucket, you can't change its Region.
* AWS Import/Export
  + Backup transfer to S3 via physical device movement
* Snowball
  + Petabyte scale transport solution
  + AWS provided secure transfer appliance
* Storage Gateway
  + **Connect on-premise data center to AWS**
  + Gateway cached volumes
    - Mount **iSCSI** devices on the on-premises servers
    - The data placed on this device will be stored in to S3. Also, frequently accessed data on-premises will be cached.
  + Gateway stored volumes
    - Data stored on the on-premises storage volume.
    - Frequently store the **snapshot data as incremental backups to s3**
* Buckets can contain both **encrypted and non-encrypted objects**

### Amazon S3 Data Consistency Model

* + Amazon **S3 provides read-after-write consistency for PUTS of new objects in** your S3 bucket in all regions with one caveat. The caveat is that if you make a **HEAD** or **GET** request to the key name (to find if the object exists) before creating the object, Amazon S3 provides **eventual consistency for** **read-after-write**.
  + Amazon S3 offers **eventual consistency for overwrite PUTS and DELETES in all regions.**
* Consistency Behaviors
  + A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.
  + A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might return the prior data.
  + A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.
  + A process deletes an existing object and immediately lists keys within its bucket. Until the deletion is fully propagated, Amazon S3 might list the deleted object.

|  |  |
| --- | --- |
| **Eventually Consistent Read** | **Consistent Read** |
| Stale reads possible | No stale reads |
| Lowest read latency | Potential higher read latency |
| Highest read throughput | Potential lower read throughput |

**Access Policy Options:**

* Resource-based policies
  + Policies that are attached to **buckets and objects**
  + Ex, **bucket policy, ACL**
  + Bucket policies are limited to **20 KB in size**.
* User policies
  + IAM policy for a user

**ACL:**

* You can use ACLs to grant basic read/write permissions to other AWS accounts.
* Limits - you can grant **permissions only to other AWS accounts**; you **cannot grant permissions to users in your account.**
* For example, if a **bucket owner allows other AWS accounts to upload objects**, permissions to these objects can only be managed **using object ACL** by the **AWS account that owns the object.**

**When to use ACL?**

Object ACL

* + If the AWS account that owns the object also owns the bucket, then it can write a bucket policy to manage the object permissions.
  + If the AWS account that owns the object wants to grant permission to a user in its account(same), it can use a user policy.

The following are the scenarios when you use object ACLs to manage object permissions.

* + - AWS account that owns the bucket can grant another AWS account permission to upload objects. **The bucket owner does not own these objects.** The AWS account that **created the object** must grant permissions using object ACLs.
    - Permissions vary by object and you need to manage permissions at the object level
      * Can grant permission for all the object key names prefix “logs\_”
      * Max 100 grants
    - Object ACLs control only object-level permissions
      * READ-ACP/WRITE-ACP
      * This limits the account to manage permissions only on specific objects by updating individual object ACLs.

Bucket ACL

* + The only recommended use case for the bucket ACL is to grant write permission to the Amazon S3 Log Delivery group to write access log objects to your bucket you want Amazon S3 to deliver access logs to your bucket, you will need to grant write permission on the bucket to the **Log Delivery group**.

**When to use Bucket policy?**

* + You want to manage cross-account permissions for all Amazon S3 permissions

Bucket policy example

Granting Permissions to Multiple Accounts with Added Conditions

x-amz-acl: public-read – AWS account which requests

{

"Version":"2012-10-17",

"Statement":[

{

"Sid":"AddCannedAcl",

"Effect":"Allow",

"Principal": {"AWS": ["arn:aws:iam::111122223333:root","arn:aws:iam::444455556666:root"]},

"Action":["s3:PutObject","s3:PutObjectAcl"],

"Resource":["arn:aws:s3:::examplebucket/\*"],

"Condition":{"StringEquals":{"s3:x-amz-acl":["public-read"]}}

}

]

}

Granting Read-only permissions to anonymous user

{

"Version":"2012-10-17",

"Statement": [

{

"Sid":"AddPerm",

"Effect":"Allow",

"Principal": **"\*",**

"Action":["s3:GetObject"],

"Resource":["arn:aws:s3:::examplebucket/\*"]

}

### Restrict Source IP address

{

"Version": "2012-10-17",

"Id": "S3PolicyId1",

"Statement": [

{

"Sid": "IPAllow",

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:\*",

"Resource": "arn:aws:s3:::examplebucket/\*",

"Condition": {

"IpAddress": {"aws:SourceIp": "54.240.143.0/24"},

"NotIpAddress": {"aws:SourceIp": "54.240.143.188/32"}

}

}

]

}

### Restrict by Referrer

* + Suppose you have a website with domain name (www.example.com or example.com) with links to photos and videos stored in your S3 bucket, **examplebucket**. By default, all the S3 resources are private, so only the AWS account that created the resources can access them. To **allow read access to these objects from your website, you can add a bucket policy that allows s3:GetObject permission with a condition**, using the **aws:Referer** key, that the get request must originate from specific webpages. The following policy specifies the **StringLike** condition with **the aws:Referer** condition key.

{

"Version":"2012-10-17",

"Id":"http referer policy example",

"Statement":[

{

"Sid":"Allow get requests originating from www.example.com and example.com.",

"Effect":"Allow",

"Principal":"\*",

"Action":"s3:GetObject",

"Resource":"arn:aws:s3:::examplebucket/\*",

"Condition":{

"**StringLike**":{"aws:Referer":["http://www.example.com/\*","http://example.com/\*"]}

}

}

]

}

**When to use User policy?**

* + Permission from the parent account
  + Permission from the resource owner

### S3 Block Public Access

* By default, new buckets and objects don't allow public access, but users can modify bucket policies or object permissions to allow public access
* You can enable Block Public Access settings only for buckets and AWS accounts. Amazon S3 doesn't support Block Public Access settings on a per-object basis.
* BlockPublicAcls
  + If TRUE, reject PUT Bucket/Object ACL calls
  + If TRUE, also block PUT object which include public ACLs
  + If TRUE at account level, it applies to all the buckets.
* IgnorePublicAcls
  + Safely block public access granted by ACLs
* BlockPublicPolicy
  + If TRUE causes Amazon S3 to reject calls to PUT Bucket policy if the specified bucket policy allows public access. This setting enables you to allow users to manage bucket policies without allowing them to publicly share the bucket or the objects it contains.
  + Best practice to enable this setting on AWS account level, to prevent users have to alter the Bucket policy also possible to disable Block public access settings.
* RestrictPublicBuckets
  + TRUE restricts access to a bucket with a public policy to only AWS services and authorized users within the bucket owner's account.
  + This setting blocks all cross-account access to the bucket

### CORS Configuration:

* A CORS preflight request is a [CORS](https://developer.mozilla.org/en-US/docs/Glossary/CORS) request that checks to see if the [CORS](https://developer.mozilla.org/en-US/docs/Glossary/CORS) protocol is understood.
* It is an [OPTIONS](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS) request, using three HTTP request headers: [Access-Control-Request-Method](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Request-Method), [Access-Control-Request-Headers](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Request-Headers), and the [Origin](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Origin) header.
* For example, a client might be asking a server if it would allow a [DELETE](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE) request, before sending a DELETE request, by using a preflight request:
  + OPTIONS /resource/foo
  + Access-Control-Request-Method: **DELETE**
  + Access-Control-Request-Headers: origin, x-requested-with
  + Origin: https://foo.bar.org

<CORSConfiguration>

<CORSRule>

<AllowedOrigin>http://www.example.com</AllowedOrigin>

<AllowedMethod>PUT</AllowedMethod>

<AllowedMethod>POST</AllowedMethod>

<AllowedMethod>DELETE</AllowedMethod>

<AllowedHeader>\*</AllowedHeader>

*<MaxAgeSeconds>3000</MaxAgeSeconds>*

*<ExposeHeader>****x-amz-server-side-encryption****</ExposeHeader>*

*<ExposeHeader>****x-amz-request-id****</ExposeHeader>*

*<ExposeHeader>****x-amz-id-2****</ExposeHeader>*

</CORSRule>

</CORSConfiguration>

### Cross-Origin Resource Sharing: Use-case Scenarios

The following are example scenarios for using CORS:

* Scenario 1: Suppose that you are hosting a website in an Amazon S3 bucket named website as described in [Hosting a Static Website on Amazon S3](https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html). Your users load the website endpoint http://website.s3-website-us-east-1.amazonaws.com. Now you want to use JavaScript on the webpages that are stored in this bucket to be able to make authenticated GET and PUT requests against the same bucket by using the Amazon S3 API endpoint for the bucket, website.s3.amazonaws.com. A browser would normally block JavaScript from allowing those requests, but with CORS you can configure your bucket to explicitly enable cross-origin requests from website.s3-website-us-east-1.amazonaws.com.
* Scenario 2: Suppose that you want to **host a web font from your S3 bucket**. Again, browsers require a CORS check (also called a preflight check) for loading web fonts. You would configure the **bucket that is hosting the web font to allow any origin to make these requests.**

### S3 CORS

Scenario 1: **Request from S3 static website to same S3 object in bucket**

* + Suppose that you are hosting a website in an Amazon S3 bucket named website as described in [Hosting a Static Website on Amazon S3](https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html). Your users load the website endpoint http://website.s3-website-us-east-1.amazonaws.com. Now you want to use JavaScript on the webpages that are stored in this bucket to be able to make authenticated GET and PUT requests against the same bucket by using the Amazon S3 API endpoint for the bucket, website.s3.amazonaws.com. A browser would normally block JavaScript from allowing those requests, but with CORS you can configure your bucket to explicitly enable cross-origin requests from website.s3-website-us-east-1.amazonaws.com.
* Scenario 2:  **Browser’s pre-flight OPTONS check for CORS** ie Access-Control-Request-Headers
  + Suppose that you want to host a web font from your S3 bucket. Again, browsers require **a CORS check (also called a preflight check) for loading web fonts**. You would configure the bucket that is hosting the web font to allow any origin to make these requests.

### Optimizing S3 Performance

* Amazon S3 automatically scales to high request rates.
* There are **no limits to the number of prefixes in a bucket**
* You can increase your read or write performance by parallelizing reads
* If you want higher transfer rates over a single HTTP connection or **single-digit millisecond** **latencies**, use [**Amazon CloudFront**](https://docs.aws.amazon.com/cloudfront/index.html)**or**[**Amazon ElastiCache**](https://docs.aws.amazon.com/elasticache/index.html) for caching with Amazon S3.
* Additionally, if you **want fast data transport over long distances** between **a client and an S3** bucket, use [**Amazon S3 Transfer Acceleration**](https://docs.aws.amazon.com/AmazonS3/latest/dev/transfer-acceleration.html)**.**
* Transfer Acceleration uses **the globally distributed edge locations in CloudFront** to accelerate data transport over geographical distances.
* Amazon S3 performance guidelines recommended **randomizing prefix naming with hashed characters** – No longer needed in newer versions
* You **no longer have to randomize prefix namin**g for performance and can use **sequential date-based naming for your prefixes**.

Guidelines

1.Measure Performance – instance types

2.Scale Storage connections – multiple concurrent requests can be issued

3.Byte range – issue concurrent requests with different Range HTTP header to single object

4.Retry requests

5.Combine S3 and EC2 in same region

6.Use latest SDK

6.Use transfer acceleration

Design Pattern

1. Use caching
2. Timeout and Retries
3. Horizontal scaling by parallel request
4. S3 Transfer acceleration

### Storage Classes

* **STANDARD**—The default storage class. If you don't specify the storage class when you upload an object, Amazon S3 assigns the STANDARD storage class.
* **REDUCED\_REDUNDANCY**—The Reduced Redundancy Storage (RRS) storage class is designed for noncritical, reproducible data that can be stored with less redundancy than the STANDARD storage class. \*\*\* AWS Recommend not to use this as this is not cost effective
* **INTELLIGENT\_TIERING** storage class is designed to optimize storage costs by automatically moving data to the most cost-effective storage access tier, without performance impact or operational overhead.
  + The INTELLIGENT\_TIERING storage class is suitable for objects larger than 128 KB that you plan to store for at least 30 days. If the size of an object is less than 128 KB, it is not eligible for auto-tiering. Smaller objects can be stored, but they are always charged at the frequent access tier rates in the INTELLIGENT\_TIERING storage class. If you delete an object before the end of the 30-day minimum storage duration period, you are charged for 30 days.
* The **STANDARD\_IA** and **ONEZONE\_IA** storage classes are designed for long-lived and infrequently accessed data.
* **GLACIER**—Use for archives where portions of the data might need to be retrieved in minutes. Data stored in the GLACIER storage class has a minimum storage duration period of 90 days and can be accessed in as little as 1-5 minutes using expedited retrieval. If you delete an object before the 90-day minimum, you are charged for 90 days. For pricing information, see [Amazon S3 Pricing](https://aws.amazon.com/s3/pricing/).

* **DEEP\_ARCHIVE**—Use for archiving data that rarely needs to be accessed. Data stored in the DEEP\_ARCHIVE storage class has a minimum storage duration period of 180 days and a default retrieval time of 12 hours. If you delete an object before the 180-day minimum, you are charged for 180 days.

When you choose the GLACIER or DEEP\_ARCHIVE storage class, your objects remain in Amazon S3. You cannot access them directly through the separate Amazon S3 Glacier service.

S3 Life Cycle management:

* Transition
* Expiration

### S3 Encryption

* **Server-side encryption**
  + SSE-S3
    - It uses **AES-256**
    - It encrypts the key and rotates the master key often
    - Request will be denied if there is no **x-amz-server-side-encryption header**
    - AWS managed
  + SSE-KMS
    - **Cost associated**
    - Uses CMKs
    - Must be used along with Sig4
    - **x-amz-server-side-encryption:"aws:kms"**
    - optionally key can be specified while making requests, if not present it assumes default KMS key. If supplied this key id should be match with the policy which allow/deny requests
      * x-amz-server-side-encryption-aws-kms-key-id header
    - S3 API **also supports encryption context**, with the x-amz-server-side-encryption-context header. This is used for **CloudTrail logging**
  + SSE-C
    - **Customer provides key as part of request**
    - **Customer manages the key**
    - Must use HTTPS
    - Following headers must be passed,
      * x-amz-copy-source​-server-side​-encryption​-customer-algorithm
      * x-amz-copy-source​-server-side​-encryption​-customer-key
      * x-amz-copy-source-​server-side​-encryption​-customer-key-MD5
    - Presigned URL and SSE-C
      * When creating a presigned URL, you must **specify the algorithm using the x-amz-server-side​-encryption​-customer-algorithm** in the signature calculation.
      * When using the presigned URL to upload a new object, retrieve an existing object, or retrieve only object metadata, you must provide **all the encryption headers** in your **client application.**
        + Can only be accessed by programmatically passing headers
* **Client-side encryption**
  + By using AWS KMS – CMK
  + By using client-side master key
  + We manage the key and we encrypt the data as well

We recommend that you use the following pattern to **encrypt data locally in your application: (ClientSide Encryption Process)**

1. Use this operation (***GenerateDataKey***) to get a data encryption key.
2. Use the **plaintext data encryption key** (returned in the Plaintext field of the response) to **encrypt data locally**, then **erase the plaintext data key from memory**.
3. **Store the encrypted data key** (returned in the CiphertextBlob field of the response) **alongside the locally encrypted data.**

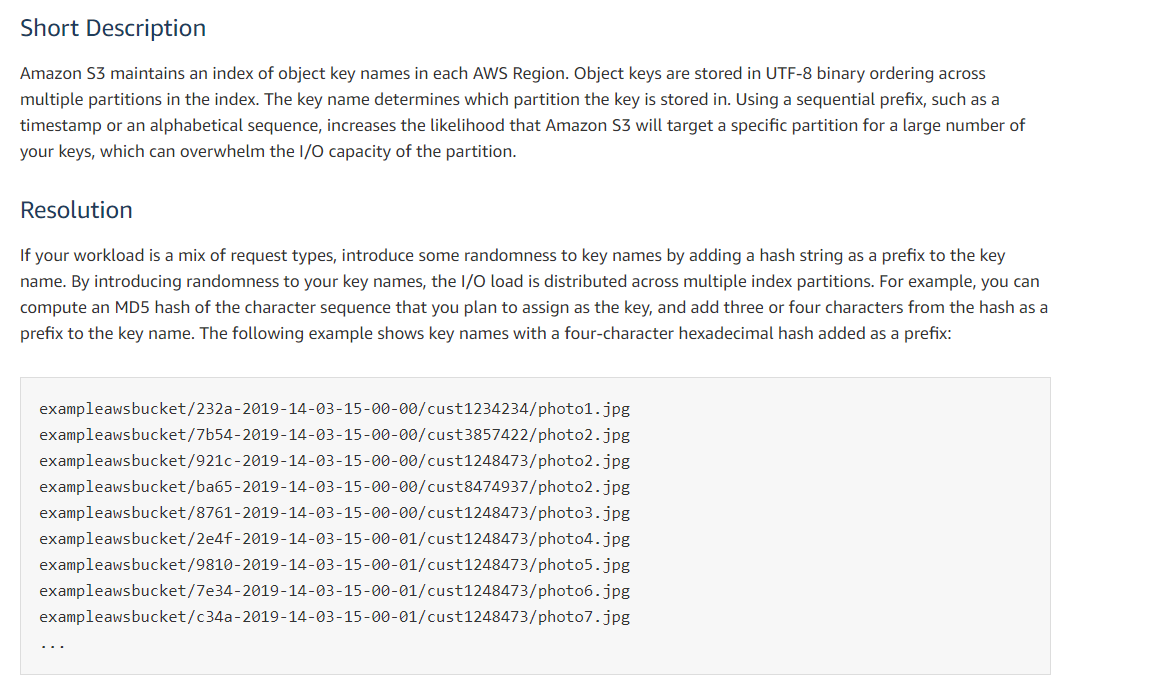
### Versioning

* Enabling and suspending versioning is **done at the bucket level.** When you enable versioning for a bucket, all objects added to it will have a unique version ID. **1024-byte long version id**
* You must explicitly enable versioning on your bucket.
* **By default, versioning is disabled**. Regardless of whether you have enabled versioning, each object in your bucket has a version ID.
* If you **have not enabled versioning, Amazon S3 sets the value of the version ID to null.**
* Pre-versioned object’s Version ID is null

Deleting versioned object

* The **delete marker becomes the current version of the object**. By default, GET requests retrieve the most recently stored version.
* Performing a simple **GET Object request when the current version is a delete marker returns a 404 Not Found error.**
* However, GET a noncurrent version of an object by specifying its version ID. In the following figure, we GET a specific object version, 111111. Amazon S3 returns **that object version even though it's not the current version.**
* You can **permanently delete an object** **by specifying the version you want to delete**.
* **Only the owner** of an **Amazon S3 bucket can permanently delete a version.**

### S3 Object keys and its’s performance



* Using sequential prefix, timestamped prefix, alphabetical sequence causes same partition used for large number of keys. Hence more I/O operation
* MD5 based hash prefix can be added

**Troubleshooting**

### Significant Increases in HTTP **503 Responses** to Amazon S3 Requests to Buckets with Versioning Enabled

If you notice a significant increase in the number of **HTTP 503-slow down responses** received for Amazon S3 PUT or DELETE object requests to a bucket **that has versioning enabled, you might have one or more objects in the bucket for which there are millions of versions.** When you have objects with millions of versions, Amazon S3 automatically throttles requests to the bucket to protect the customer from an excessive amount of request traffic, which could potentially impede other requests made to the same bucket.

To determine which S3 objects have millions of versions, **use the Amazon S3 inventory tool**. The inventory tool generates a report that provides a flat file list of the objects in a bucket.

The Amazon **S3 team encourages customers to investigate applications that repeatedly overwrite the same S3 object, potentially creating millions of versions for that object**, to determine whether the application is working as intended.

There are a number of common methods for getting your **request IDs including, S3 access logs and CloudTrail events/data events.**

### Performance Optimization

**If your workload is mainly sending GET requests**, in addition to the preceding guidelines, you should consider using **Amazon CloudFront** for performance optimization. By integrating **CloudFront with Amazon S3**, you can distribute content to your users **with low latency and a high data transfer rate**. You also s**end fewer direct requests** to Amazon S3, **which reduces your costs**.

### Prefix based

**3,500** GET per pre-fix

**5,500** GET per pre-fix

### Amazon S3 Information in CloudTrail

* **Bucket level actions tracked by CloudTrail**
  + For ex, *DeleteBucket, ListBuckets*
* **Object level actions tracked by CloudTrail**
  + For ex, *AbortMultipartUpload, DeleteObject etc*

### Amazon S3 Server Access Logging

Server access logging provides **detailed records for the requests that are made to a bucket**. Server access logs are useful for many applications. For example, access log information can be useful **in security and access audits**. It can also help you learn about your customer base and understand your Amazon S3 bill.

### S3 Pre-signed URL

* **Only the object owner has permission to access these objects**. However, the **object owner can optionally share objects with others** by creating a **pre-signed** URL, using their **own security credentials, to grant time-limited permission to download the objects**.
* Must specify
  + Bucket name
  + Object key
  + HTTP method
  + Expiration date and time

### Request Rate performance

Amazon S3 now provides increased performance to support at least **3,500 requests per second** (write) to add data and **5,500 requests per second**(read) to retrieve data.

### FAQs

How much data can I store in Amazon S3?

Min – 0 Kb, **Max – 5TB** – Object size

Single PUT – **5 GB**, **if more than 100 MB use Multipart Upload**

What is S3 Encryption Client?

You retain control of the keys and complete the **encryption and decryption of objects client-side using an encryption library** of your choice. Some customers prefer **full end-to-end control of the encryption and decryption of objects**; that way, **only encrypted objects are transmitted over the Internet to Amazon S3.** Use a client-side library if you want to maintain control of your encryption keys, are able to implement or **use a client-side encryption library and need to have your objects encrypted before they are sent to Amazon S3 for storage.**

What is an Amazon VPC Endpoint for Amazon S3?

An Amazon VPC Endpoint for Amazon S3 is a logical entity within a VPC that allows **connectivity only to S3**. The VPC Endpoint routes requests to S3 and routes responses back to the VPC

Can I allow a specific Amazon VPC Endpoint access to my Amazon S3 bucket?

You can **limit access to your bucket from a specific Amazon VPC Endpoint or a set of endpoints** using Amazon S3 bucket policies. S3 bucket policies now support a condition, **aws:sourceVpce** , that you can use to restrict access

What is Amazon S3 Object Lock?

Amazon S3 Object Lock is a new Amazon S3 feature that blocks object version deletion during a customer-defined retention period so that you can enforce retention policies as an added layer of data protection or for regulatory compliance.

 How can I configure my objects to be deleted after a specific time period?

You can set an **S3 Lifecycle expiration policy** to **remove objects from your buckets after a specified number of days.** You can define the expiration rules for a set of objects in your bucket through the Lifecycle configuration policy that you apply to the bucket.

# Cloud Formation

**Key Concepts:**

* Templates
  + .json, .yaml, .template, .txt are supported file types
* Stacks – **Max 200**
* Change Sets

Resource type identifiers always take the following form:

service-provider: service-name:data-type-name

ex, "Type": "AWS::EC2::Instance",

* CF also **supports Step Functions**
* **EB and StepFunction rely upon CF**

### Intrinsic Functions

Use intrinsic functions in your templates to assign values to properties that are not available until runtime.

You can use intrinsic **functions only in specific parts of a template**. Currently, you can use intrinsic functions **in resource properties, outputs, metadata attributes, and update policy attributes**. You can also **use intrinsic functions to conditionally create stack resources**.

* Fn::GetAtt
  + Returns **the value of attribute from a resource in the template.**
  + Ex, "Fn::GetAtt": [ "myELB" , "**DNSName**" ] – Here DNSName is attribute name. myELB – logical name
* Parameter types
  + Invalid **input for parameter values will cause stack creation failure**

"Parameters" : {

"NameOfTheParameter" : {

"Type" : "<Type Name>"

}

}

* Valid CF supported parameter types
  + **String** – A literal string
  + **Number** – An integer or float
  + **List<Number>** – An array of integers or floats
  + **CommaDelimitedList** – An array of literal strings that are separated by commas
  + **AWS::EC2::KeyPair::KeyName** – An Amazon EC2 key pair name
  + **AWS::EC2::SecurityGroup::Id** – A security group ID
  + **AWS::EC2::Subnet::Id** – A subnet ID
  + **AWS::EC2::VPC::Id** – A VPC ID
  + **List<AWS::EC2::VPC::Id>** – An array of VPC IDs
  + **List<AWS::EC2::SecurityGroup::Id>** – An array of security group IDs
  + **List<AWS::EC2::Subnet::Id>** – An array of subnet IDs
* EC2 Key pair
  + We add the EC2 key pair parameter by following

“AWS::EC2::KeyPair::KeyName”

"Parameters" : {

"KeyName": {

"Description": "Name of an existing EC2 KeyPair to enable SSH access to the instance",

"**Type": "AWS::EC2::KeyPair::KeyName**",

"ConstraintDescription": "must be the name of an existing EC2 KeyPair."

}

}

* List of EC2 security group IDs
  + The type for adding a list of EC2 Security Group IDs is “**List<AWS::EC2::SecurityGroup::Id>**”.

"Parameters" : {

"SecurityGroupIds": {

"Description": "Security groups that can be used to access the EC2 instances",

**"Type": "List<AWS::EC2::SecurityGroup::Id>",**

"ConstraintDescription": "must be list of EC2 security group ids"

}

}

* Referring the parameter

"Resources" : {

"EC2Instance" : {

"Type" : "AWS::EC2::Instance",

"Properties" : {

"InstanceType" : { "Ref" : "InstanceType" },

"ImageId" : { "Ref" : "AMIImageId" },

"SecurityGroupIds" : { "Ref" : "SecurityGroupIds" },

"KeyName" : { "Ref" : "KeyName" }

}

}

}

### Resource Attributes

* *CreationPolicy* – this is to prevent changing creation complete status until signal from AWS resource or timeout reached**. Cfn-signal** will used for signaling
* *DeletionPolicy* – **this is to preserve resources while deleting stack**
* *DependsOn* – dependent resources which needs to be created first
* *Metadata* – Add structured data to resource
* *UpdatePolicy* – to specify how CF handles updates
* *UpdateReplacePolicy* – Specify to retain resources when stack update operation. **Prevent deletion of oldresource. Can specify retain or snapshot of old resource**.

### Conditions

* Fn::And
* Fn::Equals
* Fn::If
* Fn::Not
* Fn::Or

The **CreateProdResources** condition **evaluates to true if** the **EnvType** parameter is equal to prod. In the sample template, the **NewVolume** and **MountPoint** resources are associated with the CreateProdResources condition. Therefore, the resources are created only if the EnvType parameter is equal to prod.

"Parameters" : {

"EnvType" : {

"Description" : "Environment type.",

"Default" : "test",

"Type" : "String",

"AllowedValues" : ["prod", "test"],

"ConstraintDescription" : "must specify prod or test."

}

},

"Conditions" : {

"CreateProdResources" : {"Fn::Equals" : [{"Ref" : "EnvType"}, "prod"]}

},

"MountPoint" : {

"Type" : "AWS::EC2::VolumeAttachment",

"Condition" : "CreateProdResources",

"Properties" : {

"InstanceId" : { "Ref" : "EC2Instance" },

"VolumeId" : { "Ref" : "NewVolume" },

"Device" : "/dev/sdh"

}

},

"Outputs" : {

"VolumeId" : {

"Value" : { "Ref" : "NewVolume" },

"Condition" : "CreateProdResources"

}

}

The optional Conditions section contains statements that define the circumstances under which entities are created or configured. For example, **you can create a condition and then associate it with a resource or output so that AWS CloudFormation only creates the resource or output if the condition is true**. Similarly, you can associate the condition with a property so that **AWS CloudFormation only sets the property to a specific value if the condition is true.** If the condition is false, AWS CloudFormation sets the property to a different value that you specify.

You might use conditions when you want **to reuse a template that can create resources in different contexts,** such as **a test environment versus a production environment**. In your template, you can add an EnvironmentType input parameter, which accepts either prod or test as inputs.

The AWSTemplateFormatVersion section (optional) identifies the capabilities of the template. The latest template format version is **2010-09-09** and is currently the only valid value.

### Best Practices

Planning & Organizing

* + Organize stacks
    - Managing single stack is complex
    - We can group resources based on usage without affecting other resources
      * For ex, website team, database team own two different stacks for easy maintainability. Otherwise communication and coordination is difficult.
    - Architecture
      * SOA (boundary based, own lifestyle, manageable parts, separated scope)
      * **Layered. Horizontal layered stack. Dependency layer go below**
  + Cross stack references
    - Import - **Fn::ImportValue**
    - Export
  + IAM to control access
    - **Necessary permission** must be set to work with the resources involved in the stack (Create, Update, Delete). For ex, **EC2 permissions to launch**
  + Verify quotas for all resource type
    - Ensure the resources creation are within AWS account limits
  + Reuse templates
    - We can use it for multiple environment
  + **Use nested stack to reuse common template patterns**
    - **Common resources creation can be moved to separate template** and then use it with another template in **unified stack**
    - **To create a nested stack**
      * **AWS::CloudFormation::Stack** – to refer another template
      * For example, assume that you have a load balancer configuration that you use for most of your stacks. Instead of copying and pasting the same configurations into your templates, you can create a dedicated template for the load balancer. Then, you **just use the**[**AWS::CloudFormation::Stack**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-stack.html)**resource to reference that template from within other templates.** If the load balancer template is updated, any stack that is referencing it will use the updated load balancer (only after you update the stack).

Creating templates

* + DONOT embed credentials in templates
    - **Use parameter to pass credentials**
    - Use NoEcho
  + Use AWS specific parameter types
    - For example, you can specify a parameter as type **AWS::EC2::KeyPair::KeyName, which takes an existing key pair name that is in your AWS account and in the region** where you are creating the stack
  + Use Parameter constraints
  + Use **AWS::CloudFormation::Init**
    - Software pre-configurations on EC2
  + Use latest helper scripts – **UserData**
  + **Validate templates**
    - **Always validate template for errors before you use to create or update stack**
    - If **validation fails, will not cause rollback**

Managing stacks

* + **Don’t make changes in stack resources outside of CF**. If so, update, delete stack will not work correctly.
  + **Create changesets before updating stack**
    - View **proposed changes in the stack before execution**
  + Stack policy
    - **Protect critical resources**
    - During stack update, we need to **mention protected resources explicitly otherwise no changes made to protected resources.**
  + Use CloudTrail to log CloudFormation calls
    - **Enable s3 bucket log**
  + Use CodeReviews and Revision controls to manage your templates.
    - History tracking
  + Update your EC2 Linux instance regularly.

## Stack Sets

* + AWS CloudFormation StackSets **extends the functionality of stacks by enabling you to create, update, or delete stacks across multiple accounts and regions with a single operation.**
  + Using an administrator account, you **define and manage an AWS CloudFormation template, and use the template as the basis for provisioning stacks** **into selected target accounts across specified regions**.
  + **Max 20**


            A stack set is a collection of resources, defined in a template and deployed
                into one or more accounts across one or more regions.
        

### CLI Commands

**Create stack**

aws cloudformation create-stack

**Describe and Lists**

aws cloudformation **list-stacks**

aws cloudformation **describe-stacks**

**View stack event history**

aws cloudformation **describe-stack-events**

**list-stack-resources –** Return description of all the resources in the specified stack

**describe-stack-resource** – Return description of specified resource in the specified stack

### APIs

* CreateStack
* DescribeStackResource - Return description of specified resource
* DescribeStackResources - Returns AWS resource descriptions for running and deleted stacks
* DescribeStacks – Returns description of specified stack, if no stack name given, it list all
* ListStackResources - Returns descriptions of all resources of the specified stack.

### Detecting unmanaged changes to resources outside of CF

You can **use drift detection to identify stack resources** to which configuration changes have been **made outside of CloudFormation management**. You can then take corrective action so that your stack resources are **again in sync** with their definitions in the stack template, such as updating the drifted resources directly so that they agree with their template definition.

### Helper Scripts

CF provides list of Python helper scripts that you can use to install software and start services on EC2 instance. These will not be executed by default; CF templates should invoke it.

* Cfn-init
  + Use to retrieve and interpret **resource metadata, install packages, create files, and start services**.
* Cfn-signal
  + Use to **signal with a CreationPolicy or WaitCondition,** so you can synchronize other resources in the stack when the prerequisite resource or application is ready.
* Cfn-get-metadata
  + Use to **retrieve metadata for a resource or path to a specific key**.
* Cfn-hup
  + Use to check for updates to metadata and execute custom hooks when changes are detected.

**UserData** should include **– yum install aws-cfn-bootstrap.** If not, we need manually update scripts in linux by below

**Sudo yum install -y aws-cfn-bootstrap**

### Limits

* Maximum of 100 mappings
* 64 attributes per mapping
* Max of 60 outputs
* Max of 60 parameters
* Max of 200 Resources
* Maximum 200 Stacks (Can get a increase from AWS)
* Maximum 20 StackSets
  + 500 stack instances per set

### FAQs

**Can I use AWS CloudFormation with Chef?**

**Yes**. AWS CloudFormation can be used to bootstrap both the **Chef Server and Chef Client** software on your EC2 instances.

**Can I use AWS CloudFormation with Puppet?**

**Yes**. AWS CloudFormation can be used to bootstrap both the **Puppet Master and Puppet Client software** on your EC2 instances.

**Are there limits to the number of templates or stacks?**

There are no limits to the number of templates. Each AWS CloudFormation account is limited to a maximum of **200 stacks. Complete our request for a higher limit** [here](https://aws.amazon.com/contact-us/cloudformation-request/), and we will respond to your request within two business days.

# Encryption SDK

* Client-side encryption library designed to make it easy for everyone to encrypt and decrypt data using industry standards and best practices.

Encrypt

* Up to 4 KB (4096)
* You can use the Encrypt operation to move encrypted data from one AWS region to another. In the **first region, generate a data key** and **use the plaintext key to encrypt the data**. Then, in the **new region, call the Encrypt method on same plaintext data key**. Now, you can safely move the **encrypted data and encrypted data key to the new region**, and decrypt in the new region when necessary.

IAM in X-Ray

* For X-ray console usage,
  + Must need **AWSXRayReadOnlyAccess** managed policy
* For local development and testing
  + IAM **user access keys should be used in SDK**
  + Uses the keys for **Daemon, CLI, SDK**
* For deployment app
  + **AWSXRayDaemonWriteAccess**  - to upload traces,
  + Read permissions – to sample
* For full access
  + **AWSXrayFullAccess**  - **read/write/configure encryption** key settings/sampling rules

Running your X-Ray application locally

* Daemon needs write permissions to send traces to X-Ray
* Store credentials in **~/.aws/credentials** folder for CLI to run daemon locally

When you run your application on AWS, **use a role to grant permission to the Amazon EC2 instance or Lambda function that runs the daemon.**

* EC2 – Create IAM role for **EC2 as instance profile**
* ECS – Create IAM role for Container service as **Container instance IAM role**
* EB –
  + It’s **default instance profile has the permissions for X-Ray**
  + Or add write permissions to custom instance profile
* Lambda
  + Add X-ray write permissions **to lambda execution role**
* **AmazonS3ReadOnlyAccess** (Amazon EC2 only) – Gives the instance permission to **download the X-Ray daemon** from Amazon S3.

X-Ray permissions for Encryption

* X-Ray **encrypts all trace data by default**
* If we use **AWS CMK**, then key’s policy **should allow X-Ray to use it encrypt.**

For a customer managed CMK, **configure your key with an access policy** that allows the following actions:

* User who configures the key in X-Ray has permission to call **kms:CreateGrant** and **kms:DescribeKey**.
* Users who can access encrypted trace data have permission to call **kms:Decrypt.**
* AWS Managed CMK – **aws/xray**

API Access

* **AmazonAPIGatewayInvokeFullAccess**

DynamoDB

Read/Write capacity mode

* + On demand
  + Provisioned
* You can switch between read/write capacity **modes once every 24 hours**.
* **1 RCU – 4KB**
  + **1 strong consistent read per second**
  + **2 eventually consistent read per second**
* **1 WCU – 1KB**
* **DescribeTable**  - This API call **does not consume RCU & WCU. Both 0 in this** case
* ProvisionedThroughputExceededException
  + This occurs when running out of RCU/WCU
  + Also occurs when one or more Global index is running out of RCU
  + Solution: Retries and Exponential backoff
  + Provisioned throughput and consumed throughput will be available in cloudwatch

Code Build

* Caching
  + Downloading dependencies is a critical phase in the build process. These dependent files can range in size from a few KBs to multiple MBs. Because **most of the dependent files do not change frequently between builds**, you can noticeably **reduce your build time** by **caching dependencies**.
  + **By default, cache artifact is encrypted with KMS**
  + AWS recommend you only cache dependencies/files that will not change frequently between builds

Elastic Beanstalk with S3

Elastic Beanstalk applies **a bucket policy** to buckets it creates to allow environments to write to the bucket and **prevent accidental deletion**. If you need to delete a bucket that Elastic Beanstalk created, first delete the bucket policy from the Permissions section of the bucket properties in the Amazon S3 console.

If you delete a bucket that Elastic Beanstalk created in your account, and you still have existing applications and running environments in the corresponding region, your **applications might stop working correctly**. For example:

* **When an environment scales out, Elastic Beanstalk should be able to find the environment's application version in the Amazon S3 bucket** and use it to start new Amazon EC2 instances.
* **When you create a custom platform, Elastic Beanstalk uses temporary Amazon S3 storage during the creation process.**

AppSpec ‘hooks’ section

* ECS
  + Life cycle hook
    - BeforeInstall
    - AfterInstall
    - AfterAllowTestTraffic
    - BeforeAllowTraffic
    - AfterAllowTraffic
  + Sample lambda function
    - Use the 'hooks' section to specify a Lambda function that CodeDeploy can call to validate a Lambda deployment.
    - If CodeDeploy is not notified by the Lambda validation function within one hour, then it assumes the deployment failed
* Lambda
  + Life cycle hooks
    - BeforeAllowTraffic
    - AfterAllowTraffic
* EC2
  + Life cycle
    - **ApplicationStop**
      * Runs on second time hence first-time won’t be running as the appspec not available
    - **DownloadBundle**
    - **BeforeInstall**
    - **Install**
    - **AfterInstall**
    - **ApplicationStart**
    - ApplicationStop
    - **ValidateService**
    - BeforeBlockTraffic
    - BlockTraffic
    - AferBlockTraffic
    - BeforeAllowTraffic
    - AllowTraffic
    - AfterAllowTraffic
  + Script timeout
    - **Max 3600 seconds for script to run on each deployment lifecycle event**

# Cloud Watch Alarm

* Basic Monitoring – **5 Mins or 300 seconds**
* Detailed Monitoring – **1 Minute or 60 seconds**

High Resolution alarm

* + Either 10 seconds or 30 seconds
  + This is alarm is based on High Resolution Metrics

High-Resolution Metrics

* + Standard – one **min**. granularity
  + High resolution – one **second** granularity
* **Standard resolution by default**
* CloudWatch stores it with resolution **1 Sec**, we can Read and retrieve with period of **1, 5, 30, 60 and multiple of 60 seconds**

## Evaluating an Alarm

When you create an alarm, you specify three settings to enable CloudWatch to evaluate when to change the alarm state:

* **Period** is the **length of time to evaluate the metric** or expression to **create each individual data point for an alarm.** It is expressed in seconds. If you choose one minute as the period, there is **one data point every minute**.
* **Evaluation Period(N)** is the **number of the most recent periods, or data points**, to evaluate when determining alarm state.
* **Datapoints to Alarm(M)** is the **number of data points within the evaluation period** that must be breaching to cause the alarm to go to the ALARM state. The breaching data points don't have to be consecutive, they just must all be within the last number of data points equal to **Evaluation Period**.

If M and N are different values, then its M out of N alarm.

For example, if you configure **4 out of 5 data points** with a period of 1 minute, the evaluation interval is 5 minutes. If you configure **3 out of 3** data points with a period of **10** minutes, the evaluation interval is 30 minutes.

## High-Resolution Alarms

If you set an alarm on a high-resolution metric, you can specify a high-resolution alarm with a period of 10 seconds or 30 seconds, or you can set a regular alarm with a period of any multiple of 60 seconds. There is a higher charge for high-resolution alarms. For more information about high-resolution metrics, see [Publishing Custom Metrics](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/publishingMetrics.html).

# S3 Http Error code

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Http Code** | **Description** |
| **BucketAlreadyExists** | **409 Conflict** | Specified bucket name is not available. May be in use |
| BucketAlreadyOwnedByYou | 409 Conflict | Already exists or you own it. This is except us-east-1. In us-east-1, u can re-create it |
| **BucketNotEmpty** | **409 Conflict** | The bucket is not empty while you delete |
| CrossLocationLoggingProhibited | 409 Conflict | Cross-location logging not allowed. **Buckets in one geographic location cannot log information to a bucket in another location.** |
| EntityTooSmall | 400 Bad Request | In multipart upload, minimum allowed object size 5 MB except last part. This will occur if we do multipart upload of object less that minimum size |
| InvalidBucketName | 400 Bad Request | The specified bucket name is not valid |
| **NoSuchBucket** | **404 Not Found** | Bucket does not exist |
| **NoSuchBucketPolicy** | **404 Not Found** | Bucket policy does not exist |
| **NoSuchUpload** | **404 Not Found** | **The specified multipart upload does not exist.** The **upload ID might be invalid,** or the multipart upload might have been aborted or completed. |
|  |  |  |

# Elastic Beanstalk

Deployment types

* + All at once (default)
  + Immutable
  + Rolling
  + Rolling with additional batches
  + Blue/Green

Deployment options

* + **All at once**
  + **Immutable**
    - Launch new instances side by side and do deployment
  + **Rolling**
    - Splits EC2 environments into different batches and deploy one batch at a time
  + **Rolling with Additional batches**
    - Additional batches with new version in batches **without reducing capacity**
  + Health page shows recent deployment ID of completed EC2 instance
  + **Ignore health check**
    - Option to determine whether deployment should be discarded if unhealthy instances on a group
  + **Healthy threshold** 
    - Lower the value to pass health check
  + **Command timeout**
    - Number of seconds to wait for an instance to become healthy
  + Rolling update batch size configuration
    - **.ebextensions/rolling-updates.config**

option\_settings:

aws:elasticbeanstalk:command:

DeploymentPolicy: Rolling

BatchSizeType: Percentage

BatchSize: 25

* + Rolling with additional batch config
    - To deploy to five instances in each batch, regardless of the number of instances running, and **to bring up an extra batch of five instances running the new version before pulling any instances out of service**
    - **.ebextensions/rolling-additionalbatch.config**

option\_settings:

aws:elasticbeanstalk:command:

DeploymentPolicy: RollingWithAdditionalBatch

BatchSizeType: Fixed

BatchSize: 5

**Configurations**

**Precedence**

* + Settings applied directly to the environment via console, EB CLI, AWS CLI
  + Saved configuration
  + Config. Files(.**ebextensions**)
    - **Root of the application source bundle**
  + Default values
    - Default value of specific configuration if available

.**ebextentions**

* + Location
    - All config. Files should be placed in **.ebextensions in the root of the source bundle**
  + Naming
    - Must have .config file extension
  + Formatting – JSON/YAML
  + Uniqueness – **Each key should be specified only once**

When you use the AWS Elastic Beanstalk console to deploy a new application or an application version, you'll need to upload a source bundle. Your source bundle must meet the following requirements:

* **Consist of a single ZIP** file or WAR file (you can include multiple WAR files inside your ZIP file)
* **Not exceed 512 MB**
* **Not include a parent folder or top-level directory** (subdirectories are fine)

If you want to deploy a **worker application that processes periodic background tasks**, your application source bundle must also include a **cron.yaml file**

Code Deploy Life cycle

EC2/On-Premise compute platform

* In-place
  + BeforeBlockTraffic [only on LB]
  + BlockTraffic [only on LB]
  + AfterBlockTraffic [only on LB]
  + ApplicationStop
  + DownloadBundle
  + BeforeInstall
  + Install
  + AfterInstall
  + ApplicationStart
  + ValidateService
  + BeforeAllowTraffic [only on LB]
  + AllowTraffic [only on LB]
  + AfterAllowTraffic [only on LB]
* Blue/Green
  + ApplicationStop
  + DownloadBundle
  + BeforeInstall
  + Install
  + AfterInstall
  + ApplicationStart
  + ValidateService
  + BeforeAllowTraffic
  + AllowTraffic
  + AfterAllowTraffic
  + BeforeBlockTraffic [only on original instance]
  + BlockTraffic [only on original instance]
  + AfterBlockTraffic [only on original instance]

Lambda compute platform

* BeforeAllowTraffic
* AllowTraffic
* AfterAllowTraffic

ECS Compute platform

* BeforeInstall
* Install
* AfterInstall
* AllowTestTraffic
* AfterAllowTestTraffic
* BeforeAllowTraffic
* AllowTraffic
* AfterAllowTraffic

Environment variable for hooks

* + APPLICATION\_NAME
  + **DEPLOYMENT\_ID**
  + DEPLOYMENT\_GROUP\_NAME
  + DEPLOYMENT\_GROUP\_ID
  + LIFECYCLE\_EVENT

**Customizing resources in Elastic Beanstalk**

* + **Elastic beanstalk uses Cloud Formation to create/update resources**. **All CF resource types are supported.**
  + You can use the Resources key in a [configuration file](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/ebextensions.html) to create and customize AWS resources in your environment. Resources defined in configuration files are added to the **AWS CloudFormation** **template used to launch your environment.**

# SQS

Standard Queues

* + Default queue type
  + Use cases
    - Background user work
    - Processing by worker nodes
  + Batch processing
  + Unlimited no. of transaction per second (TPS)#
  + At least once delivery

FIFO Queues

* + Batching support **3000 messages /second**
  + **300 message/second without batching**
  + Exactly once processed
  + Message deduplication ID have the value of message id and should not be processed again within 5 mins duplication interval.

Visibility Timeout

* + **Default 30 seconds**. Min – 0, Max – 12 hrs.
  + Amazon SQS sets a ***visibility timeout***, a period of time during which Amazon SQS prevents other consumers from receiving and processing the message.
  + There is no limit to the number of stored messages. (state 1 & 2)
  + A message is considered to be *in flight* after it is received from a queue by a consumer, but not yet deleted from the queue. There is limit to the number of in-flight messages. (state 2 & 3).
  + if the consumer fails before deleting the message and your system doesn't call the [DeleteMessage](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_DeleteMessage.html) action for that message **before the visibility timeout expires**, the message becomes visible to other consumers and the message is received again**. If a message must be received only once, your consumer should delete it within the duration of the visibility timeout**.
  + When receiving messages, you can also set a special **visibility timeout for the returned messages without changing the overall queue timeout.**
  + To set visibility timeout at API level while processing messages – **ChangeMessageVisibility** action
  + Set to 0 will make **message immediately available** for other consumers.
  + Strategy to provide sufficient time to process messages
    - **Heartbeat mode** by extending until your complete process
    - **Set maximum time until you** process messages

Receive request error

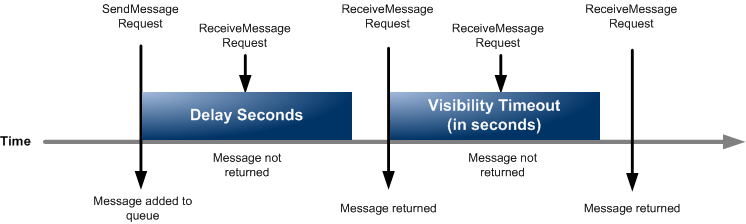
* + Allow **200 ms**(2 sec) pass between each **re-try** if not using **exponential back-off** from sdk

Long polling

* + Default **ReceiveMessage** wait time – **20 seconds**

Delay Queue

* + Default delay 0 seconds. Max **15 mins**
  + Setting this will delay in sending messages to particular queue by producers



* + Use message timer’s **DelaySeconds**  value. **FIFO don’t support delay timers**

## Processing large messages in SQS using S3

* + S3 and **SQS Extended client** library can be used
  + Useful for storing and **consuming messages up to 2 GB in size**.
  + Default message size if exceeds **over 256 KB, then consider S3**
  + SQS Extended
    - **Stored large object in S3 and reference it on a single individual message**
    - Read/Delete the related object from S3

## Limits

* + **1000 queues in** List Queues request
  + In-flight – **120000** messages for standard queue, **20000** for FIFO
  + Default message visibility timeout **– 30 seconds**. 0 to **12** hrs
  + Default receive message time – **20 seconds**.
  + Message size – **256 KB**, if exceeds S3 can be used

|  |  |
| --- | --- |
| **Standard Queue** | **FIFO Queue** |
| **Unlimited Throughput** – Standard queues support a nearly unlimited number of transactions per second (TPS) per action.  **At-Least-Once Delivery** – A message is delivered at least once, but occasionally more than one copy of a message is delivered.  **Best-Effort Ordering** – Occasionally, messages might be delivered in an order different from which they were sent. | **High Throughput** – By default, FIFO queues support up to **3,000 messages** per second **with**[**batching**](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-batch-api-actions.html)**.** **To request a limit increase,**[**file a support request**](https://console.aws.amazon.com/support/home#/case/create?issueType=service-limit-increase&limitType=service-code-sqs). FIFO queues support up to **300 messages per second, per action** (SendMessage, ReceiveMessage, or DeleteMessage) **without batching**.  **Exactly-Once Processing** – A message is delivered once and remains available until a consumer processes and deletes it. Duplicates aren't introduced into the queue.  **First-In-First-Out Delivery** – The order in which messages are sent and received is strictly preserved. |

## SQS Encryption

Scope

* **Existing messages after enabling encryption will not be encrypted**
* **Metadata will not be encrypted**
* Moving encrypted messages to unencrypted DL queue stays encrypted
* Moving unencrypted messages to encrypted DL queue stays unencrypted

Key Terms

* Data Key
  + **DEK – Data Encryption Key** responsible for **encrypting SQS contents**
* Data Key Reuse Period
  + Between 60 seconds to 24 hours (86,400)
  + Default is 300(5 mins)
  + Length of time, in seconds can re-use encrypt/decrypt before calling SQS again.
* Customer Master Key ID – KMS key**. alias/aws/sqs** – AWS managed CMK

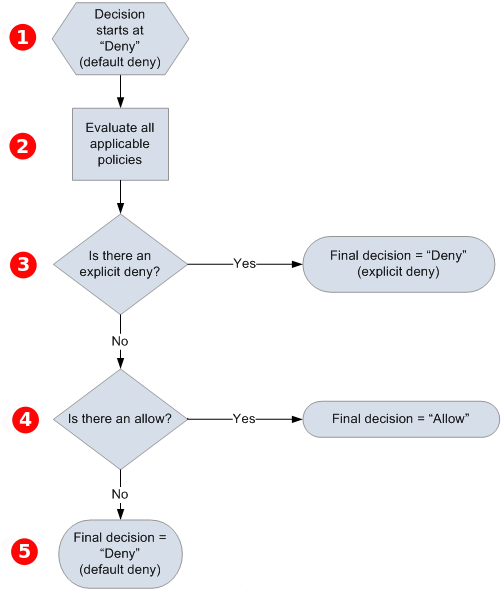
APIs

1. SendMessage
2. ReceiveMessage
   1. **Can process 10 messages at a time**
3. DeleteMessage

## SQS Access Policy Language Evaluation Logic

At evaluation time, Amazon SQS determines whether a request from someone other than the resource owner should be allowed or denied. The evaluation logic follows several basic rules:

* By default, all requests to use your resource coming from anyone but you are denied.
* An [*allow*](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-creating-custom-policies-key-concepts.html#allow) overrides any [*default-deny*](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-creating-custom-policies-key-concepts.html#default-deny).
* An [*explicit-deny*](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-creating-custom-policies-key-concepts.html#explicit-deny) overrides any **allow**.
* The order in which the policies are evaluated isn't important.



The distinction between a **default-deny** and an **explicit-deny** is important because an **allow** can overwrite the former but not the latter. For example, Policy B allows requests if they arrive on June 1, 2010. The following diagram compares combining this policy with Policy A1 and Policy A2.


      Overriding 
     

In Scenario 1, Policy A1 results in a **default-deny** and Policy B results in an **allow** because the policy allows requests that come in on June 1, 2010. The **allow** from Policy B overrides the **default-deny** from Policy A1, and the request is allowed.

In Scenario 2, Policy B2 results in an **explicit-deny** and Policy B results in an **allow**. The **explicit-deny** from Policy A2 overrides the **allow** from Policy B, and the request is denied.

## SQS FAQs

1. Maximum visibility timeout – **12 hours**
2. Max. requests in free tier – **1 million**
3. Max. number of SQS queues – **Unlimited**
4. Does SQS support anonymous queues – **Yes**?
5. Does SQS guarantee delivery of messages?
   1. **FIFO** – exactly once
   2. **Standard** – at least once
6. Batch operations cost more?
   1. No, same cost as other message request
7. Can I use **JMS (Java Message Service)** with SQS?
8. How to identify SQS message?
   1. Using **Global unique ID**
9. How to determine **time-in-queue** value?
   1. We can request ***SentTimeStamp*** attribute and subtract from current time
10. Maximum timeout for long polling? **20 seconds**
11. What is use of *AmazonSQSBufferedAsyncClient* for Java?
    1. Provides *AmazonSQSAsyncClient* interface
       1. Automatic batching
       2. Prefetch messages into local buffer for application to process
    2. Not compatible for FIFO
12. What is purge queue? – delete all messages in SQS queue without deleting queue
13. How to avoid duplicates in FIFO – introducing **message de-duplication**
14. Can I switch standard queue to FIFO? – NO
15. Are FIFO queues support multiple consumer?
    1. No for the same message group
    2. But can support different consumers for different groups
16. Message groups?
    1. Distinct ordered bundles of messages by group ID
    2. **Message group ID** is mandatory and unique
17. What happens if I issue a ***DeleteMessage*** request on a previously-deleted message?
    1. Returns success response
18. How can I enable SSE for new/existing SQS queues?
    1. Set ***KmsMasterKeyId*** in ***CreateQueue or SetQueueAttributes*** action
19. What are the things SSE does NOT do for SQS?
    1. Queue metadata
    2. Message metadata
    3. Per-queue metrics

Note: **Only Message body will be encrypted**. SSE for SQS uses **AES-GCM 256**

1. Max.number of queues? infinite

# X-Ray

**Code must be instrumented to use X-**Ray SDK (**interceptors**, **handlers**, **s**)

EC2/On-Premise

* + **Daemon should be running**
  + **IAM instance role for EC2 should be attached else AWS credentials for on-premise instance**

Lambda

* + **Make sure X-Ray integration ticked**
  + **IAM role is lambda role**When you save your function with AWS X-Ray enabled, Lambda automatically adds the permissions xray:PutTraceSegments and xray:PutTelemetryRecords to the function's current role if the function doesn't already have these permission

X-Ray on EB

* + Use beanstalk extension **(.ebextensions/xray-daemon.config**)

X-Ray on ECS/EKS/Fargate

* + Create **docker image that runs daemon**
  + Or **x-ray docker image**
  + **Port mapping/network setting**
  + **IAM task role are mapped**

X-Ray daemon/agent

* + **It has to configure to send data to account/cross account**
    - If no data, **Check IAM Permissions/Role are correct**
    - If still no data, **Check if daemon installed and running**

Segments:

Each application/service will send them

Trace:

Segments collected together to form an end-to-end trace

Sampling:

**Decrease the number of requests sent** to X-Ray

1st request in one second and 5% remining

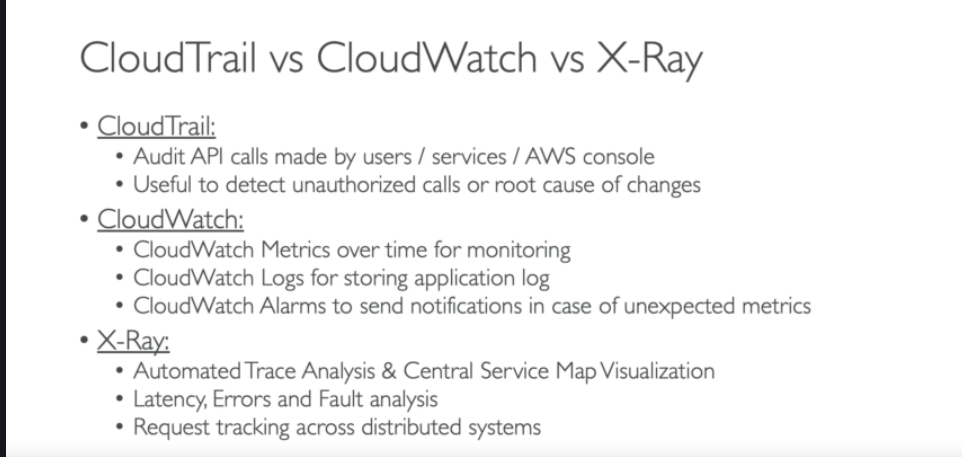
Annotations:

Key/value pairs used to index traces and use with filters

Metadata:

Key/value pairs not indexed, not searching

# CloudTrail vs CloudWatch vs X-Ray



# Cognito

* Developers can now configure a Kinesis stream to receive events as data is updated and synchronized.
* Amazon Cognito can push each dataset change to a Kinesis stream you own in real time.
* Using Amazon Cognito Streams, you can move all of your Sync data to Kinesis, **which can then be streamed to a data warehouse tool such as Amazon Redshift for further analysis.**

## User pools

1. **Sign-up and sign-in services**.
2. A built-in, **customizable web UI to sign in users**.
3. **Social sign-in with Facebook, Google, and Login with Amazon, as well as sign-in with SAML**
4. identity providers from your user pool.
5. User directory management and user profiles.
6. Security features such as multi-factor authentication (MFA), **checks for compromised**

**credential**s, account takeover protection, and phone and email verification.

1. Customized workflows and user migration through AWS Lambda triggers.

Amazon Cognito identity pools **support both authenticated and unauthenticated identities**. **Authenticated** identities belong to users who are authenticated by any supported identity provider. **Unauthenticated** identities typically belong to **guest users**.

## Cognito Stream

**Amazon Cognito Streams** gives **developers control and insight into their data stored in Amazon Cognito**. Developers can now configure **a Kinesis stream to receive events as data is updated and synchronized.**

**Amazon Cognito can** **push each dataset change to a Kinesis stream you own in real time**.

Using Amazon Cognito Streams, you can move all of your Sync data to Kinesis, which can then be streamed to a data warehouse tool such as **Amazon Redshift** for further analysis. To learn more about Kinesis, see [Getting Started Using Amazon Kinesis](https://docs.aws.amazon.com/kinesis/latest/dev/getting-started.html).

## Checking for Compromised User Credentials

* + Cognito can detect if a user’s credentials used anywhere else
  + From the **Advanced security** page in the Amazon Cognito console, you can choose whether to **allow, or block** the user if compromised credentials are detected

Events

* + Sign in
  + Sign Up
  + Password Change

## Cognito Sync

* Amazon Cognito Sync is an AWS service and client library that enable **cross-device syncing of application-related user data**.
* You can use it to **synchronize user profile data across mobile devices and web applications**. The client libraries **cache data locally so your app can read and write data regardless of device connectivity status**.
* When the **device is online, you can synchronize data, and if you set up push sync**, notify other devices immediately that an update is available.

# Kinesis KCL

KCL – Kinesis Client Library

\*\*\* Each shard is be read by only one KCL instance \*\*\*

Ex, 4 shards = max of 4 KCL instances

Progress is checkpointed into DynamoDB

\*\*\* Records are read in order at the shard level \*\*\*

Kinesis Shards

You split shards to increase the capacity (and cost) of your stream. You merge shards to reduce the cost (and capacity) of your stream.

# Lambda

Lambda function dependencies

* + Install **required packages alongside your code and zip it together**
  + If **50 MB** then zip it and upload it. Else upload to S3 and reference it

**/tmp** directory

* **512 MB max**
* **Workspace**
* Can be used for multiple invocations. Act as cache

Best practice

* + Put **heavy duty object outside of handler**
  + **Initialize aws sdk outside**
  + **Dependencies outside of handler**
  + **Sensitive env. Variable should be encrypted**

# SSM

Parameter Store supports two tiers of secure string parameters: ***standard* and *advanced***. Standard parameters, which cannot exceed 4096 bytes, are encrypted and decrypted directly under the CMK that you specify**. To encrypt and decrypt advanced secure string parameters, Parameter Store uses envelope encryption with the**[**AWS Encryption SDK**](https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/). You can convert a standard secure string parameter to an advanced parameter, but you cannot convert an advanced parameter to a standard one.

# Dynamo DB

Reserved Capacity – 100 RCU, 100 WCU

RCU

1 strongly consistent read per 4 KB

2 eventually consistent read per 4 KB (default)

2 transactional read per 4 KB

GetItem – single item

BatchGetItem – **Read upto 100 items**

* + if BatchGetItem reads a 1.5 KB item and a 6.5 KB item, DynamoDB calculates the size as 12 KB (4 KB + 8 KB), not 8 KB (1.5 KB + 6.5 KB).

Query

* + single read operation for each partition key

WCU – 1 write per 1 KB

### Hot partition

This is because before adaptive capacity, DynamoDB allocated read and write throughput evenly across partitions. For example, if you had a table capable of 400 writes per second (in other words, 400 write capacity units, or “WCUs”) distributed across four partitions, each partition would be allocated 100 WCUs. If you had a nonuniform workload with one partition receiving more than 100 writes per second (a [hot partition](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-partition-key-design.html#bp-partition-key-partitions-adaptive)), those requests might have returned a ProvisionedThroughputExceededException error.

As traffic starts to flow, **DynamoDB automatically supports your access patterns using the throughput you have provisioned**, as long as the traffic against a given partition key does not **exceed 3000 read capacity units or 1000 write capacity units**.

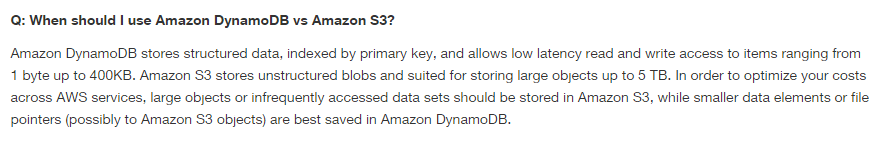
### Best practices for designing partition keys

* + Burst capacity
    - Default reservation up to 5 mins
    - When out of provisioned throughput, this will be used
  + Adaptive capacity
  + Use **partition keys to distribute work load**
  + Write sharding to distribute work load
  + **Keep the number of indexes to a minimum. Don't create secondary indexes on attributes that you don't query often.** Indexes that are seldom used contribute to increased storage and I/O costs without improving application performance.
  + **Avoid indexing tables that experience heavy write activity. In a data capture application**, for example, the cost of I/O operations required to maintain an index on a table with a very high write load can be significant. If you need to index data in such a table, it may be more effective to copy the data to another table that has the necessary indexes and query it there.

### FAQs

What is the maximum item size?

400 KB



What is the maximum number of Secondary indexes per table?

**There is an initial limit of 20 global secondary indexes per table. But you can increase this lint by requesting AWS customer support.**

A single operation can retrieve up to **16 MB of data**, which can contain as many as 100 items. **BatchGetItem** returns a partial result if the response size limit is exceeded, the table's provisioned throughput is exceeded, or an internal processing failure occurs. If a partial result is returned, the operation returns a value for **UnprocessedKeys**. You can use this value to retry the operation starting with the next item to get.

Tables per account?

**256**

But this limit can be increased by AWS

# Cloud Watch Custom metric

API call : **PutMetricData**

CLI : **put-metric-data**

\*\*\* There is no in-built metrics for **memory monitor of EC2**

Custom metrics for

the percentage of used memory, the number of total and port 80 TCP connections, the number of users currently logged in, and the percentage of I/O wait time.

#!/bin/bash

USEDMEMORY=$(free -m | awk 'NR==2{printf "%.2f\t", $3\*100/$2 }')

TCP\_CONN=$(netstat -an | wc -l)

TCP\_CONN\_PORT\_80=$(netstat -an | grep 80 | wc -l)

USERS=$(uptime |awk '{ print $6 }')

IO\_WAIT=$(iostat | awk 'NR==4 {print $5}')

**aws cloudwatch put-metric-data --metric**-name memory-usage --dimensions Instance=i-0c51f9f1213e63159 --namespace "Custom" --value $USEDMEMORY

aws cloudwatch put-metric-data --metric-name Tcp\_connections --dimensions Instance=i-0c51f9f1213e63159 --namespace "Custom" --value $TCP\_CONN

aws cloudwatch put-metric-data --metric-name TCP\_connection\_on\_port\_80 --dimensions Instance=i-0c51f9f1213e63159 --namespace "Custom" --value $TCP\_CONN\_PORT\_80

aws cloudwatch put-metric-data --metric-name No\_of\_users --dimensions Instance=i-0c51f9f1213e63159 --namespace "Custom" --value $USERS

aws cloudwatch put-metric-data --metric-name IO\_WAIT --dimensions Instance=i-0c51f9f1213e63159 --namespace "Custom" --value $IO\_WAIT

c=0

if [[ $IO\_WAIT > 70 && $USEDMEMORY > 80 ]]

then

c=1

fi

aws cloudwatch put-metric-data --metric-name danger --dimensions Instance=i-0c51f9f1213e63159 --namespace "Custom" --value $c

## Publishing the Value Zero

When your data is more sporadic and you have periods that **have no associated data**, **you can choose to publish the value zero (0) for that period or no value at all**. If you **use periodic calls to PutMetricData** to monitor the health of your application, **you might want to publish zero instead of no value.** For example, you can set a CloudWatch alarm to notify you if your application fails to publish metrics every five minutes. You want such **an application to publish zeros for periods with no associated data.**

You might also **publish zeros if you want to track the total number of data points** or if you want statistics such as minimum and average to include data points with the value 0.

# EB

Source Bundle

* + Max 512 MB
  + Don’t include parent directory on the source zip
  + If you want to **deploy worker application for background tasks, the source bundle should also include Cron.yaml**

Application version

* + Max **75 applications**
  + Max **1000 application versions**
  + Max **200 environments**

Application version lifecycle policy

* + **Useful to delete old unused application version from EB**
  + **Delete 100 versions each time when new version is deployed**, and policy applied (Excluded current version from max count)
  + Less than 10 weeks old

Monitoring environment

* + By default, only basic CloudWatch metrics are enabled, which return **5-minutes** periods. **1-min** period can be enabled.
  + Health check
    - /health url -> Http:80/health
    - No static pages should be used to check health
    - If a health check URL is configured, Elastic Load Balancing expects a GET request that it sends to return a response of 200 OK. The application fails the health check if it fails to respond within 5 seconds or if it responds with any other HTTP status code. **After 5 consecutive health check failures, Elastic Load Balancing takes the instance out of service.**

Integration with Cloud Watch

* + Elastic Beanstalk install Cloud Watch log agent on each EC2 instance that it creates.
  + Elastic Beanstalk **automatically uses Amazon CloudWatch to help you monitor your application** and environment status.
  + With CloudWatch Logs, **you can monitor and archive your Elastic Beanstalk application, system, and custom log files from Amazon EC2 instances of your environments**.
  + **The CloudWatch Logs agent installed on each Amazon EC2 instance in your environment publishes metric data points to the CloudWatch service** **for each log group you configure**.

If you can't find some of the environment's instance logs you expect in CloudWatch Logs, you can investigate the following common issues:

* **Your IAM role lacks the required IAM permissions.**
* You launched your environment in **an AWS Region that doesn't support CloudWatch Logs.**
* One of **your custom log files doesn't exist in the path you specified**.

## EB with other Services

**RDS**

* + EB can create env. With RDS database. But isn't ideal for a production environment because it ties **the lifecycle of the database instance to the lifecycle of your application's environment**.
  + To decouple your database instance from your environment, you **can run a database instance in Amazon RDS and configure your application to connect to it on launch.** This enables you to connect multiple environments to a database, **terminate an environment without affecting the database,** and perform seamless updates with blue-green deployments.
  + **For a production environment, you can**[**launch a database instance outside of your environment**](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/AWSHowTo.RDS.html)**and configure your application to connect to it outside of the functionality provided by Elastic Beanstalk**. Using a database instance that is external to your environment requires additional security group and connection string configuration. However, it also lets you connect to the database from multiple environments, use database types not supported with integrated databases, perform blue/green deployments, and tear down your environment without affecting the database instance.
  + **Enable deletion protection option on RDS instance to safeguard accidental deletion** when EB deletes environment

You can configure Elastic Beanstalk to **save a snapshot of the database when you terminate your environment**, and restore a database from a snapshot when you add a DB instance to an environment. You might **incur charges for storing database snapshots.**

The AWS documentation mentions that you should ensure the Retention field is marked as **"Create Snapshot" for saving a snapshot before the termination of the environment**. This will ensure that we can create a database from the snapshot that is created.


        Elastic Beanstalk Auto Scaling Configuration Window
      

* + RDS DB connection properties
    - RDS\_HOSTNAME
    - RDS\_PORT
    - RDS\_DB\_NAME
    - RDS\_USERNAME
    - RDS\_PASSWORD

**S3**

* + Elastic Beanstalk creates an **Amazon S3 bucket named elasticbeanstalk-*region*-*account-id***
  + It doesn’t turn **default encryption for bucket**
  + Contents that EB stores
    - **Application version** (deployable version)
    - **Source bundle.** (When delete app. Version choose “Delete version from S3” option to cleanup)
    - **Custom platforms**. Deleted after custom platform created
    - **Log files**
    - **Saved Configurations**. Manually should be removed
  + While deleting EB bucket from S3
    - **Delete “Bucket Policies” before deleting a bucket.**
    - When an **environment scales out, Elastic Beanstalk should be able to find the environment's application version in the Amazon S3 bucket** and use it to start new Amazon EC2 instances.
    - When you create a custom platform, Elastic Beanstalk uses temporary Amazon S3 storage during the creation process.

## EB issues

* + Connectivity
    - Unable to connect to Amazon RDS from Elastic Beanstalk.
  + Environment Creation
    - Failed to Launch Environment
    - Create environment operation is complete, but with command timeouts. Try increasing the timeout period.
    - The following resource(s) failed to create: [AWSEBInstanceLaunchWaitCondition]
      * If you configured your environment to launch instances in a private VPC subnet, [ensure that the subnet has a NAT](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/vpc.html) to allow the instances to connect to Elastic Beanstalk.
    - A Service Role is required in this region. Please add a Service Role option to the environment.
  + Deployments
    - Application becomes unavailable during deployments
      * Because of downtime
    - Failed to create the AWS Elastic Beanstalk application version
      * Either source bundle large or application version limit reached
    - Update environment operation is complete, but with command timeouts. Try increasing the timeout period.
      * Due to deployment time
  + Health
    - CPU Utilization Exceeds 95.00%
    - Elastic Load Balancer awseb-*myapp* Has Zero Healthy Instances
    - Elastic Load Balancer awseb-*myapp* Cannot Be Found
    - EC2 Instance Launch Failure. Waiting for a New EC2 Instance to Launch...
  + Configuration
    - You cannot configure an Elastic Beanstalk environment with values for both the Elastic Load Balancing Target option and Application Healthcheck URL option
    - ELB cannot be attached to multiple subnets in the same AZ.
  + Docker
    - ailed to pull Docker image :latest: Invalid repository name (), only [a-z0-9-\_.] are allowed. Tail the logs for more details.
    - No EXPOSE directive found in Dockerfile, abort deployment
    - Failed to download authentication credentials *repository* from *bucket name*
    - Activity execution failed, because: WARNING: Invalid auth configuration file

## Using Docker with EB

**Docker**

* Elastic Beanstalk supports the deployment of web applications from Docker containers. With Docker containers, you can define your own runtime environment. You can choose your own platform, programming language, and any application dependencies (such as package managers or tools), that aren't supported by other platforms. Docker containers are self-contained and include all the configuration information and software your web application requires to run. All environment variables defined in the Elastic Beanstalk console are passed to the containers.
* Either single or multi container based

Deployment

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.

Html

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Method** | **Impact of Failed Deployment** | **Deploy Time** | **Zero Downtime** | **No DNS Change** | **Rollback Process** | **Code Deployed To** |
| **All at once** | Downtime | Low | **☓** | ✓ | **Manual Redeploy** | Existing instances |
| **Rolling** | Single batch out of service; any successful batches before failure running new application version | Medium | ✓ | ✓ | **Manual Redeploy** | Existing instances |
| **Rolling with additional batch** | Minimal if first batch fails; otherwise, similar to **Rolling** | High | ✓ | ✓ | **Manual Redeploy** | New and existing instances |
| **Immutable** | Minimal | Very high | ✓ | ✓ | **Terminate New Instances** | New instances |
| **Blue/green** | Minimal | Very high | ✓ | **☓** | **Swap URL** | New instances |

|  |  |  |  |
| --- | --- | --- | --- |
| **Deployment Policy** | **Load-Balanced Environments** | **Single-Instance Environments** | **Legacy Windows Server Environments†** |
| All at Once | ✓ | ✓ | ✓ |
| Rolling | ✓ | ☓ | ✓ |
| Rolling with an Additional Batch | ✓ | ☓ | ☓ |
| Immutable | ✓ | ✓ | ☓ |

**Sample API URL**

**Invoke URL:** <https://gtvnj9nec1.execute-api.us-east-1.amazonaws.com/dev>

**Sample S3 Bucket URL**

Virtual hosted style

* http://*bucket*.s3.*aws-region*.amazonaws.com
* [http://*bucket*.s3.amazonaws.com](http://bucket.s3.amazonaws.com)

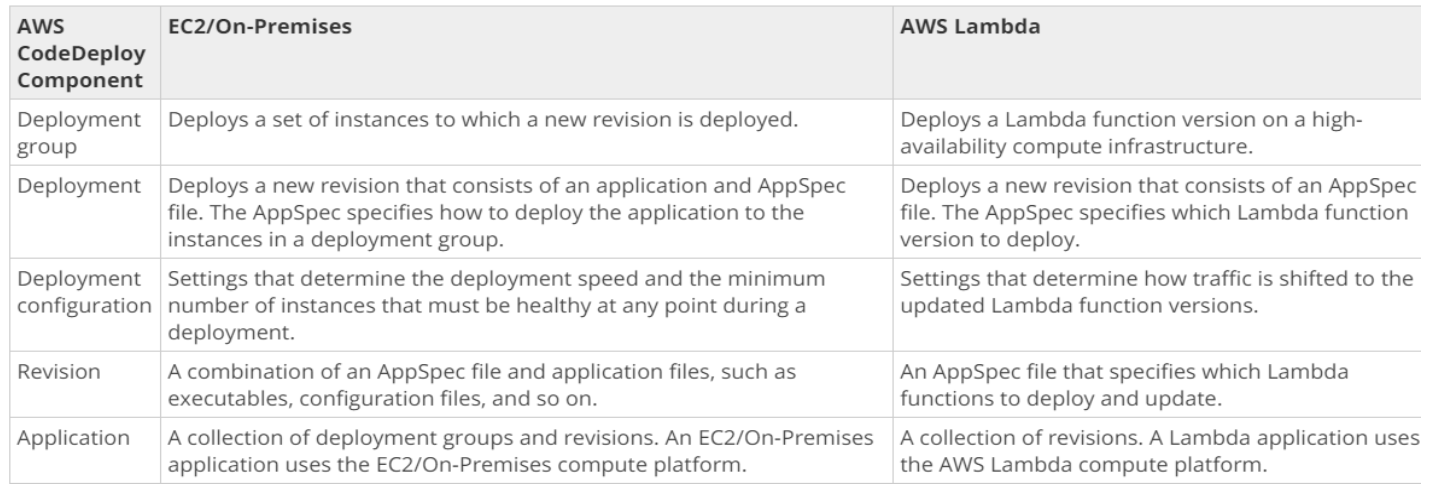
Path style

* Region-specific endpoint, http://s3.*aws-region*.amazonaws.com/*bucket*
* US East (N. Virginia) Region endpoint, http://s3.amazonaws.com/*bucket*

API Gateway cloud watch metrics

Enable Detailed CloudWatch Metrics

* + API calls, Latency, Integration Latency, 400, 500 errors



# Using Custom AMI in Elastic Beanstalk

When you create an AWS Elastic Beanstalk environment, you can specify an Amazon Machine Image (AMI) to use instead of the standard Elastic Beanstalk AMI included in your platform version. **A custom AMI can improve provisioning times when instances are launched in your environment if you need to install a lot of software that isn't included in the standard AMIs.**

Using [configuration files](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/ebextensions.html) is great for configuring and customizing your environment quickly and consistently. Applying configurations, however, can start to take a long time during environment creation and updates. **If you do a lot of server configuration in configuration files, you can reduce this time by making a custom AMI that already has the software and configuration that you need.**

A custom AMI also allows you to make changes to low-level components, such as the Linux kernel, that are difficult to implement or take a long time to apply in configuration files. To create a custom AMI, launch an Elastic Beanstalk platform AMI in Amazon EC2, customize the software and configuration to your needs, and then stop the instance and save an AMI from it.

# KMS

**Customer Master Keys**

* + AWS Owned keys (Not manageable)
  + AWS Managed keys(aws/s3)
  + Customer managed keys(cost)

**Data Keys**

* + Used to encrypt data
  + Customer master keys can be used to encrypt data key
  + AWS does not manage in KMS.

Create Data key


        Generate a data key
      

* + GenerateDataKey – API
  + GenerateDataKeyWithoutPlainText – Will return only encrypted data key
  + CMK is used
  + **Returns plaintext data key and encrypted data key** under KMS

Encrypt data with a data key


        Encrypt user data outside of AWS KMS
      

* + After using **plaintext data key to encrypt data, remove it from memory/storage**. Hence **encrypted data key along with encrypted data can be stored**.

Decrypt data with a data key

* + To decrypt data, we should pass encrypted data key to **Decrypt API** call
  + AWS KMS uses **your CMK to decrypt the data key** and then it **returns the plaintext data key**. Use **the plaintext data key to decrypt your data** and then **remove the plaintext data key from memory**


        Decrypting a data key
      

# Envelope Encryption

With Envelope Encryption, **unencrypted data is encrypted using a plaintext Data key.** This Data key is further encrypted using **plaintext Master key.** This plaintext Master key is securely stored in AWS KMS & known as **Customer Master Keys.**


        Envelope encryption
      

**Envelope encryption i**s the practice of **encrypting plaintext data with a data key**, and then encrypting the **data key under another key.**

Top level key is called as master key. KMS is to manage master keys known as CMK.


        Envelope encryption with multiple key encryption keys
      

Benefits of envelope encryption

* + **Protecting plaintext data keys**
  + Encrypting **same data under multiple master keys**
    - Instead of re-encrypting large data which causes performance issues, we re-encrypt only data keys multiple times.
  + Combine multiple algorithms
    - Can use both symmetric and public key algorithms

Encryption context

* + **Pass additional data**
  + It appears as plaintext in cloud trail logs
  + All AWS KMS cryptographic operations (the [Encrypt](https://docs.aws.amazon.com/kms/latest/APIReference/API_Encrypt.html), [Decrypt](https://docs.aws.amazon.com/kms/latest/APIReference/API_Decrypt.html), [ReEncrypt](https://docs.aws.amazon.com/kms/latest/APIReference/API_ReEncrypt.html), [GenerateDataKey](https://docs.aws.amazon.com/kms/latest/APIReference/API_GenerateDataKey.html), and [GenerateDataKeyWithoutPlaintext](https://docs.aws.amazon.com/kms/latest/APIReference/API_GenerateDataKeyWithoutPlaintext.html)) accept an encryption context, an **optional set of key–value pairs that can contain additional contextual information about the data.**

# Redis

**Gaming Leaderboards (Redis Sorted Sets)**

* **Redis sorted sets move the computational complexity associated with leaderboards from your application to your Redis cluster.**
* Leaderboards, such as the Top 10 scores for a game, are computationally complex, especially with a large number of concurrent players and continually changing scores.
* Redis sorted sets guarantee both uniqueness and element ordering. Using **Redis sorted sets,** each time a new element is added to the sorted set it's reranked in real time. It's then added to the set in its appropriate numeric position.

Ways to improve performance of application by Redis

* In-memory store
* Gaming leader boards
* Pub/Sub methodology
* Real time analytics store
  + Redis hashes

# Amazon QuickSight

**Amazon QuickSight is a business analytics service you can use to build visualizations,** perform ad hoc analysis, and get business insights from your data. It can automatically discover AWS data sources and also works with your data sources. Amazon QuickSight enables organizations to scale to hundreds of thousands of users and delivers responsive performance by using a robust in-memory engine (SPICE).

# Amazon RedShift

Data warehousing solution

# Batch Event Stream for CloudWatch Events

Using CloudWatch Events, you can monitor the progress of jobs, build AWS Batch custom workflows with complex dependencies, generate usage reports or metrics around job execution, or build your own custom dashboards. With AWS Batch and CloudWatch Events, you can eliminate scheduling and monitoring code that continuously polls AWS Batch for job status changes. Instead, handle AWS Batch job state changes asynchronously using any CloudWatch Events target, such as AWS Lambda, Amazon Simple Queue Service, Amazon Simple Notification Service, or Amazon Kinesis Data Streams.

# CloudFront Distribution

For web distributions, to **control how long your objects stay in a CloudFront cache** **before CloudFront forwards another request to your origin**, you can:

* Configure your origin to add a **Cache-Control or an Expires header field to each object.**
* Specify a value **for Minimum TTL in CloudFront cache behaviors**.
* Use **the default value of 24 hours**

## HTTPS for Communication between Viewers and CloudFront

You can configure one or more cache behaviors in your CloudFront distribution to require HTTPS for communication between **viewers and CloudFront.**

**Viewer Protocol Policy**:

* Redirect HTTP to HTTPS
* HTTPS only

**Redirect HTTP to HTTPS**

Viewers can use both protocols. HTTP GET and HEAD requests are automatically redirected to HTTPS requests. CloudFront returns **HTTP status code 301 (Moved Permanently) along with the new HTTPS URL**. The viewer then resubmits the request to CloudFront using the HTTPS URL.

**HTTPS Only**

Viewers can access your content only if they're using HTTPS. **If a viewer sends an HTTP request instead of an HTTPS request, CloudFront returns HTTP status code 403 (Forbidden) and does not return the object.**

## HTTPS for Communication between CloudFront and Custom Origin

**Origin Protocol Policy**

Change the **Origin Protocol Policy** for the applicable origins in your distribution:

* **HTTPS Only** – CloudFront uses only HTTPS to communicate with your custom origin.
* **Match Viewer** – CloudFront communicates with your custom origin using HTTP or HTTPS, depending on the protocol of the viewer request. For example, if you choose **Match Viewer** for **Origin Protocol Policy** and the viewer uses **HTTPS** to request an object from CloudFront, CloudFront also uses **HTTPS** to forward the request to your origin.

Choose **Match Viewer** only if you specify **Redirect HTTP to HTTPS** or **HTTPS Only** for **Viewer Protocol Policy**.

CloudFront caches the object only once even if viewers make requests using both HTTP and HTTPS protocols.