Contents

[Node 3](#_Toc524125443)

[Cluster 3](#_Toc524125444)

[GIT 3](#_Toc524125445)

[Spring Scheduler 3](#_Toc524125446)

[About AWS Identity and Access Management 3](#_Toc524125447)

[About Dynamo DB 3](#_Toc524125448)

[Dynamo DB Read Consistency 4](#_Toc524125449)

[Dynamo DB Data Model 4](#_Toc524125450)

[Dynamo DB APIs 4](#_Toc524125451)

[Dynamo DB Pricing & Throughput 4](#_Toc524125452)

[Dynamo DB Streams 5](#_Toc524125453)

[About EC2 5](#_Toc524125454)

[About Amazon Machine Image (AMI) 5](#_Toc524125455)

[AWS Cloudwatch 5](#_Toc524125456)

[AWS CloudTrail 5](#_Toc524125457)

[Step - How to run EC2 systems using SDK programmatically 5](#_Toc524125458)

[How EC2 is charged 5](#_Toc524125459)

[EC2 instance types 6](#_Toc524125460)

[EC2 Snapshot 6](#_Toc524125461)

[EC2 Instance Store 6](#_Toc524125462)

[Instance Metadata 6](#_Toc524125463)

[About Elastic Block Storage (EBS) 6](#_Toc524125464)

[EBS Types 7](#_Toc524125465)

[About ECS 7](#_Toc524125466)

[About Elastic File System (EFS) 7](#_Toc524125467)

[About S3 7](#_Toc524125468)

[S3 Storage Classes/Tiers 7](#_Toc524125469)

[About Spring Boot Annotation 7](#_Toc524125470)

[Creating Spring Boot Application 8](#_Toc524125471)

[Creating Docker Image from Spring Boot Application 8](#_Toc524125472)

[Push Docker Image to Amazon Web Service Elastic Container Registry (ECR) 8](#_Toc524125473)

[About AWS Elastic Container Service (ECS) 8](#_Toc524125474)

[Secure Socket Layer (SSL) 8](#_Toc524125475)

[Spring Cloud Config 9](#_Toc524125476)

[Java Heap and its Size 9](#_Toc524125477)

[Java Memory Management 9](#_Toc524125478)

[Java Heap Size and Garbage Collection 9](#_Toc524125479)

[Java Object Memory Management 9](#_Toc524125480)

[Garbage Collection Terminology 10](#_Toc524125481)

[Heap size tuning 10](#_Toc524125482)

[JSON Web Token 10](#_Toc524125483)

[Long Polling 10](#_Toc524125484)

[Non-Blocking 11](#_Toc524125485)

[Java Streams 11](#_Toc524125486)

[java.util.function package classes 11](#_Toc524125487)

[Ansible 11](#_Toc524125488)

[Database Tokenization 11](#_Toc524125489)

[Big O Analysis 12](#_Toc524125490)

[Types of Growth Rates in Big O analysis 12](#_Toc524125491)

[Sorting Algorithm Complexity 13](#_Toc524125492)

[Sorting Algorithms 13](#_Toc524125493)

## Node

A node is a device or data point in a network.

## Cluster

Computer cluster is a set of loosely/tightly connected computers that work together. They can be viewed as a single system. Node within computer clusters perform the same task.

Components of a cluster are connected to each other through LAN. Node can have its own OS.

## GIT

GIT cloning

git clone <url> --branch <branchname>

## Spring Scheduler

@EnableScheduling enables Spring's scheduled task execution capability. It is to be used on classes which has @Configuration annotation (Check About Spring Boot Annotation on @Configuration annotation). It enables to detect @Scheduled annotations on any Spring-managed beans.

Alternatively, if the class is annotated with @Component, @ComponentScan annotation can pick the class and method with @Scheduler annotation (as below).

@Configuration

@EnableScheduling

@ComponentScan(basePackages="com.myco.tasks") - MyTask class will be in com.myco.tasks package and it has a method with @Scheduled annotation.

public class AppConfig {

}

By default, Spring scheduler uses thread pool of size 1 (every task will be processed sequentially).

## About AWS Identity and Access Management

IAM consists of Users, Groups, Roles and Policy Documents. Policy documents are in JSON format which can be applied to Roles and Groups.

IAM is universal, “root account” has complete Admin Access. New users will have no permission when created, they will have Access Key ID & Secret Access Keys when first created. You can use them to access AWS via APIs/command lines.

## About Dynamo DB

Completely managed NOSQL database. AWS will help in managing the database software and provisioning of hardware for the database. AWS will automatically scale throughput capacity (amount of output that may be produced in a period of time) to meet workload demands. Dynamo DB synchronously replicates data across 3 facilities in an AWS region to enable high availability and durability (ability to withstand damage).

Dynamo DB runs on SSD for low-latency response times, high I/O performance to server high–demand workloads in low-cost price.

## Dynamo DB Read Consistency

While reading data from Dynamo DB, we have 2 categories – eventually consistent (default) and strongly consistent.

Eventually consistent – Maximizes read throughput, result might not reflect recently completed write, consistency across all copies of data happens within a second. Ideal for applications which can read after some time of update

Strongly Consistent – Result reflects all successful writes prior to the read.

## Dynamo DB Data Model

Table – Collection of data items (in relational DB, it is collection of rows). Dynamo DB is schema-less. Each table must have a primary key. Primary key can be single attribute or composite attribute key that combines 2 attributes. Primary key should be unique across every item.

Item – Consists of primary or composite key and number of attributes. Size of the data item should not exceed 400KB.

Attribute – It consists of attribute name and values (or set of values).

## Dynamo DB APIs

CreateTable, UpdateTable, DeleteTable, DescribeTable, ListTables, PutItem, BatchWriteItem (supports upto batch of 25 Put or Delete with max total request size of 16MB), UpdateItem, DeleteItem, GetItem, BatchGetItem (Supports max of 100 items with size limit of 16MB),

Query – Gets one or more items using primary key or from composite keys. Can also filter on non-key attributes. Response limit of 1 MB.

Scan – Gets all items and attributes by performing full scan across the table, can limit the return set by filtering against attributes.

## Dynamo DB Pricing & Throughput

Free Tier – 200 million requests/month (25 read/25 write capacity units), 25GB indexed storage, 2.5 million read request from Dynamo DB Streams, can deploy Dynamo DB Global Tables up to 2 regions.

One Write capacity unit – one write/second. One Read Capacity Unit – 2 reads/second (eventually consistent), 1 read/second (strongly consistent). All reads are rounded up to increments of 4 KB and writes are rounded up to increments of 1 KB.

Read Throughput (Read Capacity) = (Size of read rounded to nearest 4KB chunk/4KB) \* no of items. (Divide the result by 2 for eventually consistent throughput). If the read throughput is in decimal, round to the next higher value.

Example – Application requires to read 10 items of 1KB/second with eventual consistency.

Read Throughput = (4KB/4KB)\*10/2=5

Write throughput (Write Capacity) = Number of items \* (Size of write with unit of 1KB)

## Dynamo DB Streams

Dynamo DB Streams provides a sequence of item-level changes made to data in the tables in last 24 hours. You can access Dynamo DB stream through API. Dynamo DB streams can be read twice the rate of provisioned write capacity of your Dynamo DB table.

## About EC2

AWS Elastic Compute Cloud is a service tht provides resizable compute capacity in cloud infrastructure. Its provides the freedom to developers (:)) by providing compute control on Amazon env, reduces time to get a new server, instead focus on app development and start deploying to customized (for you) compute env.

## About Amazon Machine Image (AMI)

AMI is a packaged-up environment that includes all necessary details to setup and boot your instance. They are the units of your deployment.

AMI is regional. You can only launch AMI from the region where it is stored. But you can also copy AMIs to other regions.

## AWS Cloudwatch

Cloudwatch is monitoring service for AWS cloud resources and applications running on AWS. We can collect logs, metrics, set alarms, and react to changes in AWS resources. Standard monitoring is on 5 min basis. Detailed monitoring is on 1 min basis and charges extra.

## AWS CloudTrail

CloudTrail is for auditing.

## Step - How to run EC2 systems using SDK programmatically

You can manage compute capacity programmatically to start (RunInstances API), get status of your instance (DescribeInstancesAPI), terminate the instance (TerminateInstances API), stop the instance remaining with EBS root partition (StopInstances API), restart instances from EBS boot partition (StartInstanceAPI).

## How EC2 is charged

4 types - On-Demand, reserved, Spot and Dedicated Hosts.

On-demand - you pay per hour or second based on usage. It is recommended for users’ preferring low-cost env without up-front payment or long term commitment, applications with unpredictable workloads, testing apps for first time.

Spot - spare compute capacity in AWS cloud available in steep discounts relative to On-demand. Recommended for fault-tolerant, flexible (such as testing/dev env) apps, independent high-performance workloads etc. Pay per hour approach. If you terminate midway through the hour, you pay for the hour. If AWS terminates the spot instance, you get the hour back it was terminated.

Reserved - provide up to 75% discount compared to On-demand. Recommended for steady-state usage apps, users committing to EC2 for 1-3 years.

Dedicated Hosts - dedicated physical EC2 server. Recommended to use existing server-bound software licenses for compliance requirements. Can be purchased On-Demand or as reserved.

Per-second billing - pay for only what you use till the second of the hour. Recommended for apps such as dev/testing, data processing, batch processing.

## EC2 instance types

Grouped into 5 families - General Purpose, Compute Optimized, Memory Optimized, Storage Optimized and Accelerated Computing Instances (GoldenSCAM)

General Purpose (M5,M4,M3,T2) - suited for apps needing good memory to CPU ratios, fixed performance (M) or burst performance (T2).

Compute Optimized(C5,C4,C3) - More CPU resources than memory (RAM), suited for scale out compute intensive apps and high performance workloads.

Memory Optimized (X1e,X1,R4,R3) - larger memory sizes, suited for memory intensive applications like db and memor caching apps.

Accelerated Compute Instances (P3,P2,G3,F1) - suited for parallel processing capabilities for high performance computing and machine learning. G3 provides high performance graphics processing capabilities. F1 for FPGA (Field programmable gate array) based reconfigurable computing.

Storage Optimized (H1,I3,D2 DISH) - suited for low latency, high I/O capacity using SSD based internal storage. D2 - dense storage, H1 - HDD storage.

## EC2 Snapshot

Snapshots exist on S3. You can take a snapshot of a volume, but that will be stored on S3. They are point in time and incremental after first time (only the changes will be moved).

Snapshots of encrypted volumes are encrypted automatically. Volumes restored from encrypted snapshots are encrypted automatically. You can share only unencrypted snapshots to other AWS accounts.

Taking snapshot of root volume – Instance needs to be stopped and can be taken.

## EC2 Instance Store

It is also called Ephemeral Storage. Data on instance store is lost when underlying host fails.

## Instance Metadata

<http://169.254.169.254/latest/meta-data> Metadata url to get all instance details.

## About Elastic Block Storage (EBS)

Instead of storing the data on local instance store, data stored on Amazon EBS can persist independently of the life of instance. If you are using EBS as root partition, set the Delete on Termination flag as “No” if you want EBS to persist outside of the instance.

You cannot mount 1 EBS volume to multiple EC2 instances. Root volumes cannot be encrypted, but additional volumes can be. In order to encrypt root volumes, you can use third party vendor software.

## EBS Types

2 Major categories – SSD for transactional workloads (performance in IOPS) and HDD for throughput workloads (performance in MB/s). SSD can be used as root volumes.

Provision IOPS SSD – More than 10,000 IOPS

General Purpose SSD - Up to 10,000 IOPS

Throughput Optimized HDD – for data warehousing, frequently accessed workloads

Cold HDD – less frequently accessed data

Magnetic HDD – cheap, infrequently accessed storage and can be used as root volume.

## About ECS

ECS is a highly scalable, high performance container mgmt. service - like managing cluster of EC2 instances. AWS ECS makes it easy to use containers for installation, operation and scaling. It lets us to schedule long-running applications, services and batch process.

There are no costs for ECS. You pay for AWS resources created to store and running the application. ECS supports Docker containers currently.

## About Elastic File System (EFS)

EFS is a fully managed service to setup and scale file storage within AWS cloud. It provides file system interface, consistent and file-locking capabilities, concurrently accessible storage for multiple EC2 instances. EFS uses NFSv4.1 protocol. Data is stored across multiple Availability Zones within a region.

Read After Write Consistency – Once you put the object in it, you will be able to read from it (which S3 also provides).

## About S3

S3 is object based storage where we can upload files (not suited of OS, softwares etc which can be stored on EBS). Files can be from 0 to 5TB. S3 is universal namespace, bucket names must be unique globally. S3 consists of Key (name of object), Value (data), Version ID, Metadata, Access Control Lists.

Read after Write Consistency for PUTS of new objects. Eventual Consistency for overwrite PUTS and DELETES (can take some time for update and delete).

S3 is good versioning tool. Once versioning is enabled, it cannot be disabled, but suspended. During cross region replication, versioning needs to be enabled for source and destination bucket.

## S3 Storage Classes/Tiers

S3 – durable, immediately available, frequently accessed

S3 – IA – durable, immediately available, infrequently accessed

S3 – Reduced Redundancy Storage – data afford to lose, easily reproducible

Glacier – For Archived data, takes 3-5 hours for retrieval

## About Spring Boot Annotation

@SpringBootApplication - annotation that needs to be added to base class of SpringBoot class. It consists of

@Configuration : tags the class as source of bean definitions for app context.

@EnableAutoConfiguration : tells Spring boot to start adding beans based on classpath.

SpringBoot automatically add @EnableWebMvc if it sees spring-webmvc in classpath. This will activate DispatcherServlet for the web application.

@ComponentScan - tells Spring to look for components, configurations and services within the defined package.

## Creating Spring Boot Application

Go to https://spring.io/guides/gs/spring-boot/.

git clone https://github.com/spring-guides/gs-spring-boot.git

Go to pom.xml directory and run

mvn package && java -jar target/gs-spring-boot-0.1.0.jar

## Creating Docker Image from Spring Boot Application

Creating docker Image

docker build -f Dockerfile -t gs-spring-boot .

Listing Docker Images

docker images

Running Docker

docker run -p 8085:8085 gs-spring-boot

Listing docker containers

docker ps –a

## Push Docker Image to Amazon Web Service Elastic Container Registry (ECR)

aws ecr create-repository --repository-name {name}

Output of above command will have the repositoryUri which will be used in following commands.

docker tag {imagename} {repositoryUri}

aws ecr get-login --no-include-email - Command to get docker login authentication string for ECR

Output of above command will provide docker login command which you need to run.

docker push {repositoryUri} - Command to push docker image to ECR.

## About AWS Elastic Container Service (ECS)

In AWS ECS, hierarchy level is container, task, service and cluster (topmost).

Task is like a blueprint - ECS will know from Task which docker image should be used for containers, how many containers to be used for tasks and resource allocation of containers.

Service has the data of how many copies of task definition to run and maintain in a cluster. We can use ELB for ECS to maintain the number of tasks. We can also use Autoscaling to adjust the number of tasks.

## Secure Socket Layer (SSL)

SSL provides secure communication over the network. It uses cryptographic process and provides enhancement to TCP/IP protocol. Data communication between client and server over SSL starts with handshake. SSL session first decides on cipher suite (cipher is nothing but cryptographic algorithms and key sizes) that will be used by client and server. Client and server can authenticate to each other before exchanging data. After the authentication is data, data is exchanged between client and server based on agreed upon cryptographic hash function.

## Spring Cloud Config

Spring Cloud Config provides server and client side support for externalized configuration in a distributed system. With the Config Server you have a central place to manage external properties for applications across all environments.

## Java Heap and its Size

Java Virtual machine is a “virtual” execution engine instance that executes the bytecodes in Java class on a microprocessor. Java heap is where the objects of a Java program live. It is the repository of live, dead objects and some free memory.

-Xmnsize – Initial and max size of heap for young generation (new objects). Oracle recommends to keep the size between a half and a quarter overall heap size.

-Xmssize – Initial size of the heap (must be multiple of 1024 and greater than 2 MB).

-Xmxsize – Max size of the heap

## Java Memory Management

Prior to Java 8, memory in Java was split into heap and non-heap space, In Java, we have Objects and Classes (Objects are instantiations of Classes). Object data is stored on heap space, class data is stored on non-heap space (permanent generation or PermGen).

Class data will consist of name of class, methods, fields, object arrays and type arrays associated with class. PermGen also has JVM’s internal objects and compiler information.

Java 8 replaces PermGen with MetaSpace. PermGen used to be in touch with heap data, but MetaSpace is held in native memory.

## Java Heap Size and Garbage Collection

When an object can no longer be reached from any pointer in the running program, it is considered to be garbage collected. One of the best practice is to tune the time spent doing the garbage collection to within 5% of execution time.

JVM heap size determines how often VM spends on garbage collection. Setting a large heap size makes garbage collection slower and less frequent. Optimum way is to set heap size in accordance to your memory needs.

Goal of tuning heap size is to minimize JVM spending time on garbage collection while increasing the number of clients served.

## Java Object Memory Management

Every Java object in memory has 2 areas: header area and data area. Header area contains details like pointer to object class, garbage collection status, hash code, lock fields, age information, length of the array etc. Data area contains the values of all instance variables. Header area layout is fixed for a particular JVM implementation (e.g. uses 2/3 machine words (3 words if an array, 1 word is 4 bytes in 32 bit architecture)). Data area is dependent on the object layout.

## Garbage Collection Terminology

* Dangling reference is a reference in memory that is already reclaimed but still being referenced by a program. Example, two modules m1 and m2 uses reference r1, but m2 accidentally reclaims the memory referenced by r1, but m1 can try to use r1.
* Memory leak is a reference in memory where the reference is not used but not reclaimed. Example, two modules m1 and m2 uses reference r1, and both of them don’t reuse them, but never reclaim the memory. Excessive memory leaks will make the program run out of memory.
* Dead object or garbage is an object that cannot be used by running program in future. Opposite is live object.

## Heap size tuning

Determination of proper JVM heap size can be done by considering below parameters.

* How many different applications will be deployed to single JVM process (number of EAR/WAR/JAR)
* How many Java classes will be potentially loaded at runtime, including third party APIs
* Estimation of in-memory caching footprint (internal cache structures defined by third party APIs and cached data from database/file)
* Estimation of number of threads used by application

## JSON Web Token

JSON web Token is an open standard to transmit information securely between parties as JSON object. JSON object can be verified and trusted as it is digitally signed (using secret key or private/public key). It can be used during authentication (Single Sign On today uses JWT where JWT is passed on subsequent request to allow the user to access allowed resources) and information exchange (secure transformation as digitally signed).

It consists of header.payload.signature.

Header consists of type and algorithm used. Eg : { “alg”:”HS256”,”type”:”JWT” }. Then, it is encoded in Base64Url.

Payload contains the claims (statements about an entity and its metadata). There are 3 types of claims: Registered, Public and Private. Registered is a set of predefined claims which are optional like iss (issuer), exp(expiration time), aud(audience). Public claims can be defined at will using JWTs. Private claims are custom claims to share information between parties. Payload will also be encoded in Base64Url.

Signature can be created by taking the encoded header, encoded payload, a secret key, algorithm mentioned in the header and digitally sign it.

## Long Polling

HTTP Long polling is a technique where the client polls the data for new information (basically sends a request). The server holds the request until new data is available. Once the data is available, server responds back. The client will send another request and this goes on.

## Non-Blocking

With the increase of requests to server-side due to Ajax, long polling and streaming, if the servers continue to maintain blocking thread for each request, memory footprint and cost of context switching between threads becomes expensive. Non-blocking request processing is to address this problem where instead of one blocking thread per request, we will have a small number of threads to handle large number of requests asynchronously.

## Java Streams

Streams in java is a sequence of elements from a source to support aggregate operations. Streams don’t actually store elements, they provide an interface to a sequence of values from source (collections, arrays, I/O resources) and they compute over these values on demand.

Streams enable pipelining, since it returns stream object except collect and we can do complex computes one by one with this capability. Streams do internal iteration as opposed to external iteration in collections. Stream operations can be categorized as intermediate (return type is stream) and terminal operations (closing stream like collect). Intermediate operations do not perform any processing until a terminal operation is invoked (by default, *lazy*).

Example: List<Integer> transactionsIds =

transactions.parallelStream()

.filter(t -> t.getType() == Transaction.GROCERY)

.sorted(comparing(Transaction::getValue).reversed())

.map(Transaction::getId)

.collect(toList());

## java.util.function package classes

These are functional interfaces which has only one abstract method. Functional interfaces within this package follow naming convention as:

* Basic function shapes: Function (from T to R), Consumer (from T to void), Predicate (from T to Boolean), Supplier (from null to R). Some functions extends these basic function shapes, like BiFunction (T and U to R).

## Ansible

IT Automation tool to configure systems, deploy software and orchestrate IT tasks such as continuous deployments or rolling updates.

## Database Tokenization

Tokenization is the process of substituting a sensitive data element with non-sensitive value, referred as token and the actual data being transmitted or stored is not usable without an appropriate decoding. Token is a reference that maps back to sensitive data through tokenization system. It is better to have tokenization system completely isolated from data processing systems and applications that previously processed or stored sensitive data.

Tokenization is different from encryption. Both are cryptographic data security methods, but tokenization is a non-mathematical approach to replace sensitive data with non-sensitive substitutes without altering the type or length of data. Encryption actually changes the data length and type and will provide information in unreadable format.

## Big O Analysis

Big O analysis is a measure of efficiency of the algorithm by the time it takes to run as a function of its input size. In big-O analysis, figure out what the input is (n) and express the number of operations the algorithm performs in terms of n, eliminate all but the higher-order terms and remove all constant factors.

Fastest possible running time for any run-time analysis is O(1), referred as constant running time which always takes the same amount of time to execute regardless of input size.

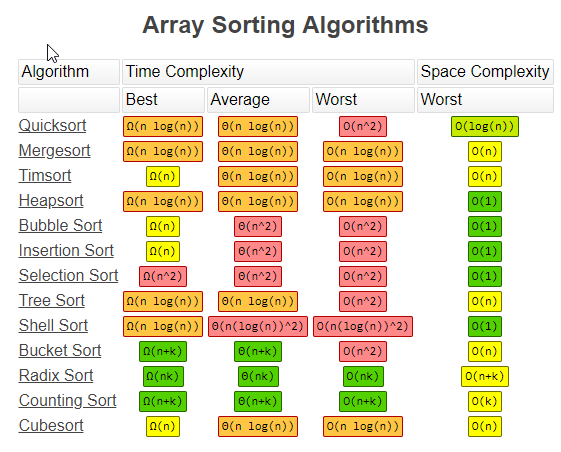
Big O analysis can change based on best, average and worst cases. That’s why we need to consider multiple scenarios before we decide an algorithm for a logic. By default, Big O analysis talks about time complexity when it measures in terms of time taken. Big O analysis can also be done with respect to space complexity which measures the amount of memory an algorithm needs to store as a function of its input size.

## Types of Growth Rates in Big O analysis

Efficiency is from above to below.

* Constant – 1
* Logarithmic – log n
* Linear – n
* Linearithmic – n log n
* Quadratic – n2
* Cubic – n3
* Polynomial – nc
* Exponential – 2n
* Factorial – n!

## Sorting Algorithm Complexity



## Sorting Algorithms

Insertion Sort: Array is logically divided into 2 parts: sorted sub-array and unsorted sub-array. Initially first element is in sorted sub-array and all other elements are in unsorted sub-array. In each iteration, move left most element in unsorted sub-array to sorted sub-array. Insertion sort is online sorting algorithm (can sort as it receives it) meaning it is a fit case for a situation where elements are coming and we need to maintain a sorted array.

Sorting Algorithms Used in Java API

Arrays.sort uses Quicksort for primitive data types and it uses Mergesort for comparison of objects.

Jenkins

Jenkins is an open source automation server which can be used to automate tasks relating to building, testing and deploying applications. Jenkins support “master/agent (slave)” mode where the workload are delegated to multiple “agent” nodes.

Creating Slaves in Jenkins

GC Objects Generation

Java garbage collector is referred to as a *Generational Garbage Collector.* Objects live in an application for varying lengths of time depending on where they are created and how they are used.

Young generation: vast majority of objects are allocated to young generation and when it fills up, it causes *minor collection* in which only young generation is collected.

Tenured generation: Fraction of young generation objects moved to tenured generation during each minor collection. When tenured generation fills up, it causes *major collection* and it last much longer than minor collection.

GC Performance Considerations

Throughput: percentage of total time not spent in GC considered over long periods of time, it includes time spent in allocation.

Pauses: times when an application appears unresponsive because GC is occurring.

Footprint: working set of a process measured in pages and cache lines.

Promptness: time between when an object becomes dead and memory becomes available.

GC CPU Limit Reached

Open the heap dump in Eclipse Memory Analyzer.