**Kafka:**

[**https://www.javainuse.com/misc/apache-kafka-hello-world**](https://www.javainuse.com/misc/apache-kafka-hello-world)

**Project discussions:**

Migrated the product module from Request-Response Architecture to Event-Driven Microservices Architecture.

OutBox Pattern:

<https://medium.com/contino-engineering/publishing-events-to-kafka-using-a-outbox-pattern-867a48e29d35>

Design a video recommendation engine to give recommendation of last 7 days top videos

<https://medium.com/contino-engineering/publishing-events-to-kafka-using-a-outbox-pattern-867a48e29d35>

Sol:

Client -🡪 AssetService --🡪 WorkFlowService --🡪 KnowledgeCatalogService

**Akka:**

[**https://www.youtube.com/watch?v=xddHqIcnvHw**](https://www.youtube.com/watch?v=xddHqIcnvHw)

**Book on Kafka**

[**https://ru.b-ok.as/book/11760445/f7b66f?dsource=recommend**](https://ru.b-ok.as/book/11760445/f7b66f?dsource=recommend)

**Kafka:**

[**https://github.com/eugenp/tutorials**](https://github.com/eugenp/tutorials)

**Spring controller:**

[**https://github.com/lokeshgupta1981/Spring-Boot-RestTemplate/blob/main/spring-demo-webapp/src/main/java/com/howtodoinjava/app/controller/UserController.java**](https://github.com/lokeshgupta1981/Spring-Boot-RestTemplate/blob/main/spring-demo-webapp/src/main/java/com/howtodoinjava/app/controller/UserController.java)

[**https://www.baeldung.com/kafka-connectors-guide**](https://www.baeldung.com/kafka-connectors-guide)**????**

**CQRS:**

[**https://david-romero.github.io/articles/2018-10/cqrs-with-kafka-streams**](https://david-romero.github.io/articles/2018-10/cqrs-with-kafka-streams)

**DataStructures:**

**Backtraing:**

[**https://leetcode.com/problems/subsets-ii/submissions/**](https://leetcode.com/problems/subsets-ii/submissions/)

[**https://leetcode.com/problems/letter-case-permutation/discuss/115485/Java-Easy-BFS-DFS-solution-with-explanation**](https://leetcode.com/problems/letter-case-permutation/discuss/115485/Java-Easy-BFS-DFS-solution-with-explanation)

[**https://leetcode.com/problems/remove-invalid-parentheses/discuss/75027/Easy-Short-Concise-and-Fast-Java-DFS-3-ms-solution**](https://leetcode.com/problems/remove-invalid-parentheses/discuss/75027/Easy-Short-Concise-and-Fast-Java-DFS-3-ms-solution)

[**https://leetcode.com/problems/time-needed-to-inform-all-employees/discuss/532560/JavaC%2B%2BPython-DFS**](https://leetcode.com/problems/time-needed-to-inform-all-employees/discuss/532560/JavaC%2B%2BPython-DFS)

**Low level deisgn:**

[**https://github.com/prasadgujar/low-level-design-primer/blob/master/questions.md**](https://github.com/prasadgujar/low-level-design-primer/blob/master/questions.md)

1. **cricBuzz**

**Multithreading:**

[**https://github.com/urvi2095/multithreading-projects-java/blob/master/Auction\_Service/Project2\_Guidelines.pdf**](https://github.com/urvi2095/multithreading-projects-java/blob/master/Auction_Service/Project2_Guidelines.pdf)

[**https://pathikritghosh.medium.com/building-a-generic-data-pipeline-with-flink-and-kafka-558d709f68b4**](https://pathikritghosh.medium.com/building-a-generic-data-pipeline-with-flink-and-kafka-558d709f68b4)

[**https://pathikritghosh.medium.com/building-a-generic-data-pipeline-with-flink-and-kafka-558d709f68b4**](https://pathikritghosh.medium.com/building-a-generic-data-pipeline-with-flink-and-kafka-558d709f68b4)

[**https://pathikritghosh.medium.com/building-a-generic-data-pipeline-with-flink-and-kafka-558d709f68b4**](https://pathikritghosh.medium.com/building-a-generic-data-pipeline-with-flink-and-kafka-558d709f68b4)

[**https://medium.com/things-solver/streaming-analytics-in-banking-how-to-start-with-apache-flink-and-kafka-in-7-steps-2171a0c5379e**](https://medium.com/things-solver/streaming-analytics-in-banking-how-to-start-with-apache-flink-and-kafka-in-7-steps-2171a0c5379e)

[**https://medium.com/@bala.dutt/streaming-system-tutorial-with-flink-and-kafka-9c445e4daa6c**](https://medium.com/@bala.dutt/streaming-system-tutorial-with-flink-and-kafka-9c445e4daa6c)

[**https://medium.com/@bala.dutt/streaming-system-tutorial-with-flink-and-kafka-9c445e4daa6c**](https://medium.com/@bala.dutt/streaming-system-tutorial-with-flink-and-kafka-9c445e4daa6c)

**Kappa Architecture**

**(Telecom: fraud detection)**

**AWS lambda:**

[**https://livebook.manning.com/book/aws-lambda-in-action/chapter-2/**](https://livebook.manning.com/book/aws-lambda-in-action/chapter-2/)

**AWS lambda example git:**

[**https://github.com/serverless/examples**](https://github.com/serverless/examples)

**Apache Flink:**

[**https://medium.com/analytics-vidhya/building-an-apache-flink-application-from-scratch-7d12ce00de8d**](https://medium.com/analytics-vidhya/building-an-apache-flink-application-from-scratch-7d12ce00de8d)

[**https://github.com/liorksh/FlinkBasicDemo**](https://github.com/liorksh/FlinkBasicDemo)

[**https://www.youtube.com/watch?v=\_8fHV5woDtQ**](https://www.youtube.com/watch?v=_8fHV5woDtQ)

**inflix DB : timeseries database**

* **Flinks windows API:**
* **Allowed lateness delay example**

**ETl with apache flink example : git**

[**https://github.com/rmetzger/flink-streaming-etl/blob/master/src/main/java/com/dataartisans/StreamingETL.java**](https://github.com/rmetzger/flink-streaming-etl/blob/master/src/main/java/com/dataartisans/StreamingETL.java)

**>>>**

[**https://github.com/mbsambangi/aws-java-spring-cloud-function-demo**](https://github.com/mbsambangi/aws-java-spring-cloud-function-demo)

**Kafka Event (git hub):**

[**https://github.com/djarza/football-events**](https://github.com/djarza/football-events)

**Apache serde: for serialization and deserialization**

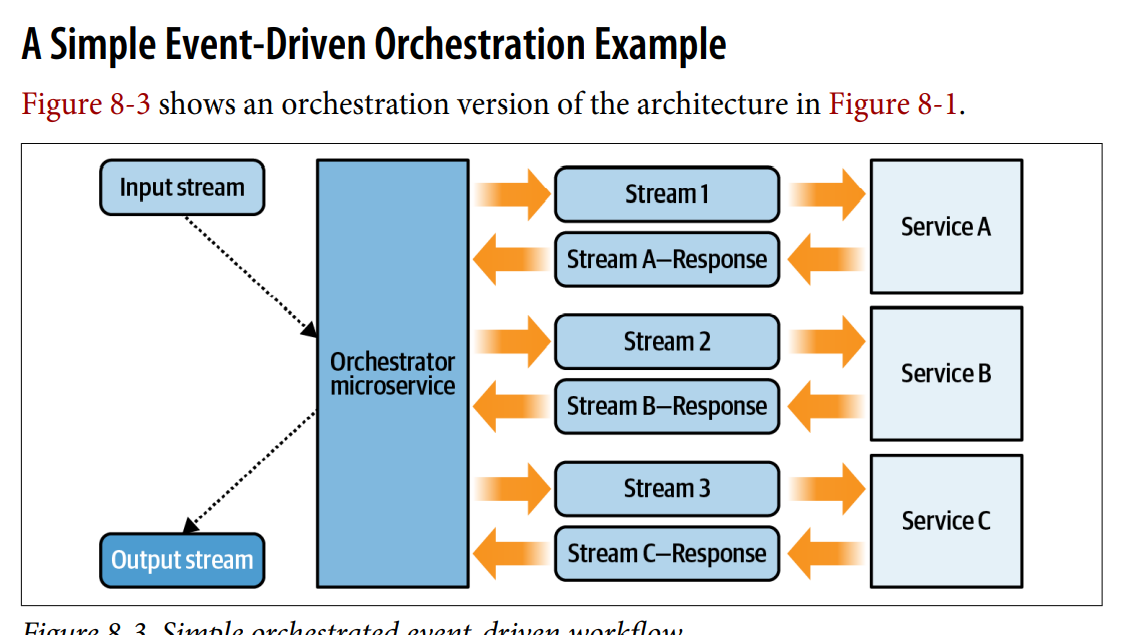
**ProcessorContext an annotation example of evolving interface**

**Orchstrator Microservices Event-dirven**

**Function as a service(FaaS) example**

**Deploy microservices as FaaS in AWS Lambda(To do)**

**WebHooks**



**Converted Direct call Orchestration to Event-Driven Orchestration(in resume)**

**Kafka Stream:**

[**https://www.youtube.com/watch?v=rqjdSbIOrJ4**](https://www.youtube.com/watch?v=rqjdSbIOrJ4)

**Event driven architecture:**

[**https://www.youtube.com/watch?v=gIL8rW\_eyww**](https://www.youtube.com/watch?v=gIL8rW_eyww)

[**https://www.youtube.com/watch?v=gIL8rW\_eyww**](https://www.youtube.com/watch?v=gIL8rW_eyww)

**To do:**

**Cqrs:**

[**https://github.com/fuinorg/ddd-cqrs-4-java-example.git**](https://github.com/fuinorg/ddd-cqrs-4-java-example.git)

**High Level Design:**

***DB Sharding: 1. Primary key(Hash Value)***

**Job Scheduler:**

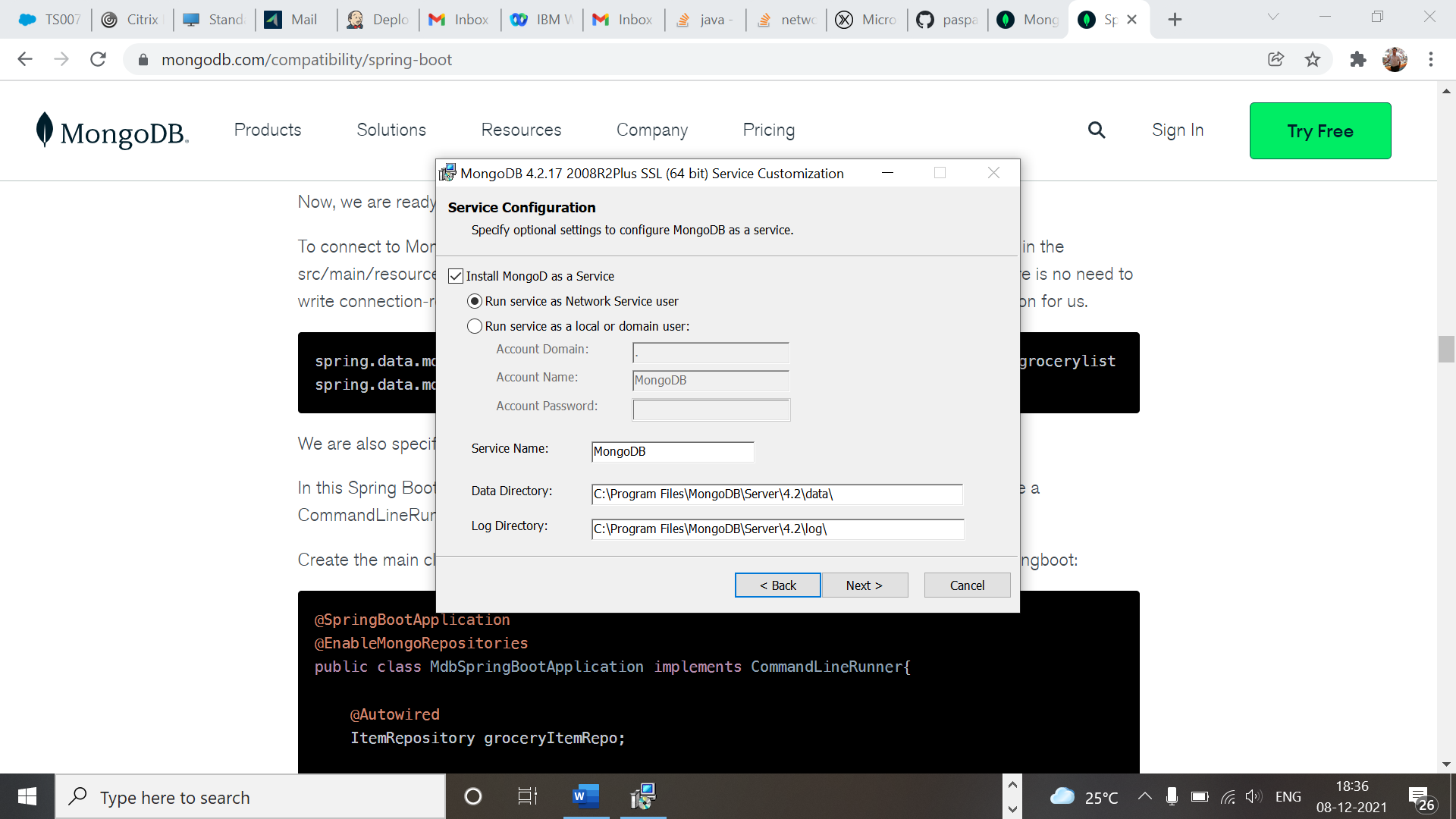
[**https://leetcode.com/discuss/general-discussion/1082786/System-Design%3A-Designing-a-distributed-Job-Scheduler-or-Many-interesting-concepts-to-learn**](https://leetcode.com/discuss/general-discussion/1082786/System-Design%3A-Designing-a-distributed-Job-Scheduler-or-Many-interesting-concepts-to-learn)

**SpringBoot with MongoDB**

Spring Boot creates quick production-ready applications. MongoDB and Spring Boot interact using the [MongoTemplate](https://docs.spring.io/spring-data/mongodb/docs/current/api/org/springframework/data/mongodb/core/MongoTemplate.html" \t "_target) class and [MongoRepository](https://docs.spring.io/spring-data/mongodb/docs/current/api/org/springframework/data/mongodb/repository/MongoRepository.html" \t "_target) interface.

* MongoTemplate — MongoTemplate implements a set of ready-to-use APIs. A good choice for operations like update, aggregations, and others, MongoTemplate offers finer control over custom queries.
* MongoRepository — MongoRepository is used for basic queries that involve all or many fields of the document. Examples include data creation, viewing documents, and more.

Spring Boot MongoDB configuration using both approaches needs only a few lines of code.



**MongoDb springBoot:**

**Saga using spring and Kafka:**

[**https://itnext.io/microservices-architecture-an-implementation-of-saga-pattern-dbbe5d881c78**](https://itnext.io/microservices-architecture-an-implementation-of-saga-pattern-dbbe5d881c78)

[**https://github.com/paspao/McPaspaoTakeAway**](https://github.com/paspao/McPaspaoTakeAway)

**simply with mongoDB:**

[**https://github.com/TechPrimers/spring-boot-mongodb-example**](https://github.com/TechPrimers/spring-boot-mongodb-example)

**CQRS:**

[**https://medium.com/@berkaybasoz/event-sourcing-and-cqrs-with-axon-and-spring-boot-part-1-6d1c1d4d054e**](https://medium.com/@berkaybasoz/event-sourcing-and-cqrs-with-axon-and-spring-boot-part-1-6d1c1d4d054e)

**Saga impl:**

[**https://github.com/Java-Techie-jt/saga-choreography-example/blob/main/saga-choreography-pattern/common-dtos/src/main/java/com/javatechie/saga/commons/event/OrderEvent.java**](https://github.com/Java-Techie-jt/saga-choreography-example/blob/main/saga-choreography-pattern/common-dtos/src/main/java/com/javatechie/saga/commons/event/OrderEvent.java)

[**https://morioh.com/p/44d546f8310c**](https://morioh.com/p/44d546f8310c)

[**https://axoniq.io/resources/concepts**](https://axoniq.io/resources/concepts) **Va Apache camel**

**Kakfa topic:**

**Using programmatic Kafka topic:**

[**https://howtodoinjava.com/kafka/multiple-consumers-example/**](https://howtodoinjava.com/kafka/multiple-consumers-example/)

[**https://github.com/lokeshgupta1981/Kafka-Tutorials/blob/master/spring-boot-kafka-app/src/main/java/com/howtodoinjava/kafka/demo/SpringBootKafkaAppApplication.java**](https://github.com/lokeshgupta1981/Kafka-Tutorials/blob/master/spring-boot-kafka-app/src/main/java/com/howtodoinjava/kafka/demo/SpringBootKafkaAppApplication.java)

[**https://codenotfound.com/spring-kafka-consumer-producer-example.html**](https://codenotfound.com/spring-kafka-consumer-producer-example.html)

[**https://www.confluent.io/blog/apache-kafka-spring-boot-application/#five**](https://www.confluent.io/blog/apache-kafka-spring-boot-application/#five)

[**https://www.javainuse.com/misc/apache-kafka-hello-world**](https://www.javainuse.com/misc/apache-kafka-hello-world)

**MicroServices:**

[**https://github.com/JohnChangUK/Payment-Choreography-Saga**](https://github.com/JohnChangUK/Payment-Choreography-Saga)

[**https://medium.com/@cheron.antoine/reactor-java-1-how-to-create-mono-and-flux-471c505fa158**](https://medium.com/@cheron.antoine/reactor-java-1-how-to-create-mono-and-flux-471c505fa158)

**Flux and Mono**

[**https://www.javainuse.com/misc/apache-kafka-hello-world**](https://www.javainuse.com/misc/apache-kafka-hello-world)

**InfirmationServer:**

[**https://www.ibm.com/docs/en/iis/11.7?topic=components-tier-relationships**](https://www.ibm.com/docs/en/iis/11.7?topic=components-tier-relationships)

**Migration from monolith to microservices:**

[**https://www.youtube.com/watch?v=d\_wVZ6sgVbs**](https://www.youtube.com/watch?v=d_wVZ6sgVbs)

**1.**

**Convert 2D to 1D in DP**

**2. determine the time complexity**

**3. Bits**

**3d array:**

[**https://leetcode.com/problems/minimum-cost-to-merge-stones/discuss/247567/JavaC%2B%2BPython-DP**](https://leetcode.com/problems/minimum-cost-to-merge-stones/discuss/247567/JavaC%2B%2BPython-DP)

[**https://leetcode.com/problems/maximal-rectangle/discuss/29054/Share-my-DP-solution**](https://leetcode.com/problems/maximal-rectangle/discuss/29054/Share-my-DP-solution)

[**https://leetcode.com/problems/split-array-largest-sum/discuss/89817/Clear-Explanation%3A-8ms-Binary-Search-Java**](https://leetcode.com/problems/split-array-largest-sum/discuss/89817/Clear-Explanation%3A-8ms-Binary-Search-Java)

[**https://leetcode.com/problems/combination-sum-iv/**](https://leetcode.com/problems/combination-sum-iv/)

[**https://leetcode.com/problems/house-robber-ii/**](https://leetcode.com/problems/house-robber-ii/)

[**https://leetcode.com/problems/combination-sum-ii/**](https://leetcode.com/problems/combination-sum-ii/)

[**https://leetcode.com/problems/permutations-ii/**](https://leetcode.com/problems/permutations-ii/)

[**https://leetcode.com/problems/binary-tree-maximum-path-sum/**](https://leetcode.com/problems/binary-tree-maximum-path-sum/)

**https://leetcode.com/problems/word-search-ii/(Trie)**

[**https://leetcode.com/problems/combination-sum/discuss/16502/A-general-approach-to-backtracking-questions-in-Java-(Subsets-Permutations-Combination-Sum-Palindrome-Partitioning)**](https://leetcode.com/problems/combination-sum/discuss/16502/A-general-approach-to-backtracking-questions-in-Java-(Subsets-Permutations-Combination-Sum-Palindrome-Partitioning))

[**https://leetcode.com/problems/combination-sum-iii/**](https://leetcode.com/problems/combination-sum-iii/)

[**https://leetcode.com/problems/target-sum/**](https://leetcode.com/problems/target-sum/)

[**https://leetcode.com/problems/n-queens/discuss/19805/My-easy-understanding-Java-Solution**](https://leetcode.com/problems/n-queens/discuss/19805/My-easy-understanding-Java-Solution)

[**https://leetcode.com/problems/permutations-ii/**](https://leetcode.com/problems/permutations-ii/)

[**https://leetcode.com/problems/reorder-list/**](https://leetcode.com/problems/reorder-list/)

[**https://leetcode.com/problems/reorder-list/**](https://leetcode.com/problems/reorder-list/)

[**https://leetcode.com/problems/insert-interval/discuss/21602/Short-and-straight-forward-Java-solution**](https://leetcode.com/problems/insert-interval/discuss/21602/Short-and-straight-forward-Java-solution)

[**https://leetcode.com/problems/minimum-number-of-refueling-stops/**](https://leetcode.com/problems/minimum-number-of-refueling-stops/)

[**https://leetcode.com/problems/last-stone-weight-ii/**](https://leetcode.com/problems/last-stone-weight-ii/)

[**https://leetcode.com/problems/minimum-cost-for-tickets/**](https://leetcode.com/problems/minimum-cost-for-tickets/)

[**https://leetcode.com/problems/distinct-subsequences/**](https://leetcode.com/problems/distinct-subsequences/)

[**https://leetcode.com/problems/shortest-common-supersequence/**](https://leetcode.com/problems/shortest-common-supersequence/)

**System Design:**

**1.**[**https://medium.com/airbnb-engineering**](https://medium.com/airbnb-engineering)

**2. Amazon Architecture:**

[**http://highscalability.com/amazon-architecture**](http://highscalability.com/amazon-architecture)

**DS:**

**Must to do:**

**1.**

[**https://leetcode.com/problems/sliding-window-maximum/**](https://leetcode.com/problems/sliding-window-maximum/)

[**https://leetcode.com/problems/course-schedule/**](https://leetcode.com/problems/course-schedule/)

[**https://leetcode.com/discuss/interview-question/448285/List-of-questions-sorted-by-common-patterns**](https://leetcode.com/discuss/interview-question/448285/List-of-questions-sorted-by-common-patterns)

**Paymanent gateway:**

[**https://www.muvi.com/blogs/understanding-payment-gateway.html**](https://www.muvi.com/blogs/understanding-payment-gateway.html)

[**https://leetcode.com/discuss/career/452354/compilation-of-questions-covering-each-interview-question-pattern**](https://leetcode.com/discuss/career/452354/compilation-of-questions-covering-each-interview-question-pattern)

<https://leetcode.com/problems/max-consecutive-ones-iii/discuss/247564/JavaC%2B%2BPython-Sliding-Window>

**Sliding window**

**Arrays :**

1. Subarray with given sum
2. Kadane’s Algorithm
3. Missing number in array
4. Merge two sorted arrays
5. Rearrange array alternatively
6. Number of pairs
7. Inversion of Array
8. Sort an array of 0s, 1s and 2s
9. Equilibrium point in array
10. Leaders in an array
11. Minimum Platforms
12. Reverse array in groups
13. K’th smallest element
14. Trapping Rain Water/ Histogram
15. Pythagorean Triplet
16. Chocolate Distribution Problem
17. Stock buy and sell
18. Element with left side smaller and right side greater
19. Convert array into Zig-Zag fashion
20. Last Index of 1
21. Spirally traversing a matrix
22. Largest Number formed from an Array

**String :**

1. Reverse words in a given string
2. Permutations of a given string
3. Longest Palindrome in a String
4. Recursively remove all adjacent duplicates
5. Check if string is rotated by two places
6. Roman Number to Integer
7. Anagram
8. Remove Duplicates
9. Form a Palindrome
10. Longest Distinct Characters in the string
11. Implement Atoi
12. Implement strstr
13. Longest Common Prefix

**Linked List :**

1. Finding middle element in a linked list
2. Reverse a linked list
3. Rotate a Linked List
4. Reverse a Linked List in groups of given size
5. Intersection point in Y shaped linked lists
6. Detect Loop in linked list
7. Remove loop in Linked List
8. n’th node from end of linked list
9. Flattening a Linked List
10. Merge two sorted linked lists
11. Intersection point of two Linked Lists
12. Pairwise swap of a linked list
13. Add two numbers represented by linked lists
14. Check if Linked List is Palindrome
15. Implement Queue using Linked List
16. Implement Stack using Linked List
17. Given a linked list of 0s, 1s and 2s, sort it
18. Delete without head pointer

**Stack and Queue :**

1. Parenthesis Checker
2. Next larger element
3. Queue using two Stacks//  implemented using one user stack and one Function Call Stack
4. Stack using two queues
5. Get minimum element from stack
6. LRU Cache
7. Circular tour
8. First non-repeating character in a stream
9. Rotten Oranges
10. Maximum of all subarrays of size k

**Tree :**

1. Print Left View of Binary Tree
2. Check for BST
3. Print Bottom View of Binary Tree
4. Print a Binary Tree in Vertical Order
5. Level order traversal in spiral form
6. Connect Nodes at Same Level
7. Lowest Common Ancestor in a BST
8. Convert a given Binary Tree to Doubly Linked List
9. Write Code to Determine if Two Trees are Identical or Not
10. Given a binary tree, check whether it is a mirror of itself
11. Height of Binary Tree
12. Maximum Path Sum
13. Diameter of a Binary Tree
14. Number of leaf nodes
15. Check if given Binary Tree is Height Balanced or Not
16. Serialize and Deserialize a Binary Tree

**Heap :**

1. Find median in a stream
2. Heap Sort
3. Operations on Binary Min Heap
4. Rearrange characters
5. Kth largest element in a stream
6. Merge K sorted linked lists
7. Kth largest element in a stream

**Recursion :**

1. Flood fill Algorithm
2. Number of paths in matrix
3. Combination Sum
4. Josephus problem

**Hashing :**

1. Relative Sorting
2. Sorting Elements of an Array by Frequency
3. Largest subarray with 0 sum
4. Common elements
5. Find all four sum numbers
6. Swapping pairs make sum equal
7. Count distinct elements in every window
8. Array Pair Sum Divisibility Problem
9. Longest consecutive subsequence
10. Array Subset of another array
11. Find all pairs with a given sum
12. Find first repeated character
13. Zero Sum Subarrays
14. Minimum indexed character
15. Check if two arrays are equal or not
16. Uncommon characters
17. Smallest window in a string containing all the characters of another string
18. First element to occur k times
19. Check if frequencies can be equal

**Graph :**

1. Depth First Traversal
2. Breadth First Traversal
3. Detect cycle in undirected graph
4. Detect cycle in a directed graph
5. Topological sort
6. Find the number of islands
7. Implementing Dijkstra
8. Minimum Swaps
9. Strongly Connected Components
10. Shortest Source to Destination Path
11. Find whether path exist
12. Minimum Cost Path
13. Circle of Strings
14. Floyd Warshall
15. Alien Dictionary
16. Snake and Ladder Problem

**Greedy :**

1. Activity Selection
2. N meetings in one room
3. Coin Piles
4. Maximize Toys
5. Page Faults in LRU
6. Largest number possible
7. Minimize the heights
8. Minimize the sum of product
9. Huffman Decoding
10. Minimum Spanning Tree
11. Shop in Candy Store
12. Geek collects the balls

**Dynamic Programming :**

1. Minimum Operations
2. Max length chain
3. Minimum number of Coins
4. Longest Common Substring
5. Longest Increasing Subsequence
6. Longest Common Subsequence
7. 0 – 1 Knapsack Problem
8. Maximum sum increasing subsequence
9. Minimum number of jumps
10. Edit Distance
11. Coin Change Problem
12. Subset Sum Problem
13. Box Stacking
14. Rod Cutting
15. Path in Matrix
16. Minimum sum partition
17. Count number of ways to cover a distance
18. Egg Dropping Puzzle
19. Optimal Strategy for a Game
20. Shortest Common Supersequence

**Divide and Conquer :**

1. Find the element that appears once in sorted array
2. Search in a Rotated Array
3. Binary Search
4. Sum of Middle Elements of two sorted arrays
5. Quick Sort
6. Merge Sort
7. K-th element of two sorted Arrays

**Backtracking :**

1. N-Queen Problem
2. Solve the Sudoku
3. Rat in a Maze Problem
4. Word Boggle
5. Generate IP Addresses

**Bit Magic :**

1. Find first set bit
2. Rightmost different bit
3. Check whether K-th bit is set or not
4. Toggle bits given range
5. Set kth bit
6. Power of 2
7. Bit Difference
8. Rotate Bits
9. Swap all odd and even bits
10. Count total set bits
11. Longest Consecutive 1’s
12. Sparse Number
13. Alone in a couple
14. Maximum subset XOR

**Java:**

Illegal Exception in iterating stream

<https://www.journaldev.com/2389/java-8-features-with-examples>

**GateWay:**

[**https://github.com/TechPrimers/spring-cloud-gateway-example-1**](https://github.com/TechPrimers/spring-cloud-gateway-example-1)

**Circuit breaker:**

[**https://howtodoinjava.com/spring-cloud/spring-hystrix-circuit-breaker-tutorial/**](https://howtodoinjava.com/spring-cloud/spring-hystrix-circuit-breaker-tutorial/)

**Load balancer:**

[**https://www.studytonight.com/post/load-balancing-spring-boot-microservices-using-netflixs-ribbon**](https://www.studytonight.com/post/load-balancing-spring-boot-microservices-using-netflixs-ribbon)

**Saga Pattern:**

[**https://vinsguru.medium.com/choreography-saga-pattern-with-spring-boot-microservice-design-patterns-fb35a1802bee**](https://vinsguru.medium.com/choreography-saga-pattern-with-spring-boot-microservice-design-patterns-fb35a1802bee)

**Using Orchestrator:**

[**https://github.com/vinsguru/vinsguru-blog-code-samples/blob/master/architectural-pattern/saga-orchestration/order-orchestrator/src/main/java/com/vinsguru/saga/config/WebClientConfig.java**](https://github.com/vinsguru/vinsguru-blog-code-samples/blob/master/architectural-pattern/saga-orchestration/order-orchestrator/src/main/java/com/vinsguru/saga/config/WebClientConfig.java)

**java8:**

[**https://dzone.com/articles/reactor-core-tutorial**](https://dzone.com/articles/reactor-core-tutorial)

**Microservices examples:**

[**https://github.com/vinsguru/vinsguru-blog-code-samples**](https://github.com/vinsguru/vinsguru-blog-code-samples)

**SideCar Pattern**

**CQRS using event(queue) and**

[**https://www.vinsguru.com/cqrs-pattern-microservice-design-patterns/**](https://www.vinsguru.com/cqrs-pattern-microservice-design-patterns/)

reactor.core.publisher.Sinks ??

**System Design(leetcode)**

**1.**

1. Design a transaction history and how you will implement it. Interviewer is focusing on the api request, response and HTTP method used and asked related questions to it.

**Resolution for ISOLATION property in SAGA:**

1. **Version File**

**SAGA implementation:**

[**https://github.com/piomin/sample-spring-boot-saga-eventing/blob/master/product-saga/src/main/java/pl/piomin/samples/saga/product/ProductSagaApplication.java**](https://github.com/piomin/sample-spring-boot-saga-eventing/blob/master/product-saga/src/main/java/pl/piomin/samples/saga/product/ProductSagaApplication.java)

**Microservices design patterns code:**

[**https://github.com/wuyichen24/microservices-patterns**](https://github.com/wuyichen24/microservices-patterns)

**Why do sub-sequence approach in REST is NOT stateless?**

**gRPC Vs REST**

**RPI proxy for circuit breaker.**

**Hystrix as library to implement circuit breaker algo.(TO DO)**

**Message Relay?? How it works to poll messages from the queue?**

[**https://medium.com/@narengowda/system-design-dropbox-or-google-drive-8fd5da0ce55b**](https://medium.com/@narengowda/system-design-dropbox-or-google-drive-8fd5da0ce55b)

**Micorservices tutorial:**

[**https://www.youtube.com/watch?v=Z7A\_M8HkJG0&t=1s**](https://www.youtube.com/watch?v=Z7A_M8HkJG0&t=1s)

**(Complete)**

1. **Kafka Asynchronous send example?TO DO**
2. **Avro example to compress the data**
3. **How do Cumsumer defines the Offset**
4. **Why do Async and Sync commit ?**
5. **Avro Deserialization?**

**Microservices Architecture pattern:**

**RPC?**

**gRPC vs REST as communication protocol**

**remote procedure invocation protocol**

**(Make project in gRPC)**

Using the Exception tracking pattern

**Resiliance4j example:**

[**https://reflectoring.io/retry-with-springboot-resilience4j/**](https://reflectoring.io/retry-with-springboot-resilience4j/)

**API Composition in API Gateway(TO DO)**

**: request routing**

**ServiceMesh:**

**Istio ServiceMesh Impl.**

[**https://www.youtube.com/watch?v=nnxWMhy0mpA**](https://www.youtube.com/watch?v=nnxWMhy0mpA)

**Microservices chahssis(for tracing)**

**Service Mesh:**

**Spring Cloud:**

[**https://developer.okta.com/blog/2020/12/07/spring-cloud-config**](https://developer.okta.com/blog/2020/12/07/spring-cloud-config)

**MicroServices Notes:**

[**https://medium.com/edureka/microservices-design-patterns-50640c7bf4a9**](https://medium.com/edureka/microservices-design-patterns-50640c7bf4a9)

Hexagonal Architecture

**1.**

**Aggregator Pattern**

[**https://java-design-patterns.com/patterns/aggregator-microservices/**](https://java-design-patterns.com/patterns/aggregator-microservices/)

**2.**

**API Gateway**

**3.**

**Chained OR Chain of Responsibility Pattern**

**4.**

**ASynchronous Messaging Design Pattern**

**5. DB or Shared Data pattern**

**6.Event Sourcing Design Pattern**

**7. Branch Pattern**

**8. CQRS: Command Query Responsibility Segregator**

**9. Circuit Breaker**

**10. Decomposition pattern**

**Microservices using spring:**

[**https://www.youtube.com/c/TechPrimers/playlists**](https://www.youtube.com/c/TechPrimers/playlists)

[**https://medium.com/edureka/microservices-design-patterns-50640c7bf4a9**](https://medium.com/edureka/microservices-design-patterns-50640c7bf4a9)

**Microservices pattern:**

[**https://microservices.io/articles/applying.html**](https://microservices.io/articles/applying.html)

[**https://microservices.io/patterns/data/cqrs.html**](https://microservices.io/patterns/data/cqrs.html)

**KAFKA:**

[**https://github.com/ewolff/microservice-kafka/blob/master/microservice-kafka/microservice-kafka-invoicing/src/main/java/com/ewolff/microservice/invoicing/events/OrderKafkaListener.java**](https://github.com/ewolff/microservice-kafka/blob/master/microservice-kafka/microservice-kafka-invoicing/src/main/java/com/ewolff/microservice/invoicing/events/OrderKafkaListener.java)

**Database scaling:**

[**https://medium.com/swlh/5-database-scaling-solutions-you-need-to-know-e307570efb72**](https://medium.com/swlh/5-database-scaling-solutions-you-need-to-know-e307570efb72)

**REST API:**

[**https://medium.com/javarevisited/rest-api-best-practices-with-design-examples-from-java-and-spring-web-services-ebec6161ccd1**](https://medium.com/javarevisited/rest-api-best-practices-with-design-examples-from-java-and-spring-web-services-ebec6161ccd1)

The typical frameworks that are used to build RESTful web services in the Java world, are Spring MVC, Spring REST and JAX-RS.

Each resource can have multiple representations — XML or JSON format

Always use plurals when you name resources

Another way you can do versioning is by specifying a header.

Here, we use a header named X-API-VERSION, and have labeled the URI as /person/header. When the header value is 1, the resource of type PersonV1

We have so far seen four types of versioning techniques:

* URI versioning
* Request Param versioning
* Header versioning
* Media Type versioning
* **URI Pollution** : With URI versioning and request parameter versioning, we end up polluting the URI space. This is because we add prefixes and suffixes to the core URI strings. Header versioning avoids that.
* **Misuse Of HTTP Headers**: In the case of Header versioning and Media Type versioning, there is a misuse of HTTP headers, since they were not originally meant for versioning.
* **Caching**: A resource is defined by its URI. However, if you are not using the URI to determine its version, but using a header-based mechanism, the versioning information cannot be cached. If HTTP caching is important to you, use the URI or Request Parameter versioning.
* **Browser Request Executability**: Both Header and Media Type versioning need the use of tools such as Postman to execute the browser request. However, if the consumers of the service are not technically well versed, then URI or Request Parameter versioning would be preferable.
* **API Documentation**: You also need to think about how you want to document your APIs. The URI and Request Parameter versioning are easier to document than the other two versioning types.
* 200 — Success
* 404 — Resource Not Found
* 400 — Bad Request (such as validation error)
* 201 — Created
* 401 — Unauthorized (when authorization fails)
* 500 — Server Error

## **Error Details In Response Body**

It helps if you have a standard exception structure when designing your service.

**1.**

**https://itzone.com.vn/en/article/5-scaling-database-solutions/**

* Cache , **DB index** so the search time is logn as its BST
* Session ID; store as key,value in Table in DB but can be huge ; **secondly in In memory DB, using JWT so no need for web server side session management**
* With server-side sessions, you will either have to store the session identifier in a database, or else keep it in memory and make sure that the client always hits the same server. Both of these have drawbacks. In the case of the database (or other centralised storage), this becomes a bottleneck and a thing to maintain - essentially an extra query to be done with every request.
* Example of **JWTToken???**
* **Master-Slave architecture; defer reading writing to different machines**
* Database sharding: vertical/horizontal

**IIS is an** integrated software platform; consisting of set of core functional module sthat enables to integrate data from disparate resources . considerd in 3 logical tiers:

1. platform foundation 2. Layer of Information Oriented Servcies 3.Series of optional modules

**1**

1.1 database 1.2. WAS

2

2.1 Metadata Management

2.2 Connectivity

2.3 Administration

3.

3.1 Data Profiling

3.2 Transform

**Microservices patterns:**

**Chain Pattern,**

<https://medium.com/geekculture/design-patterns-for-microservices-5362689581c4>

**SpringBoot:**

[**https://github.com/subhashlamba/spring-boot-microservice-example/tree/main/spring-boot-cloud-api-gateway-routing**](https://github.com/subhashlamba/spring-boot-microservice-example/tree/main/spring-boot-cloud-api-gateway-routing)

[**https://github.com/OmarElGabry/microservices-spring-boot**](https://github.com/OmarElGabry/microservices-spring-boot)

**(important)**

**RDBMS:**

Leetcode design:

Design custom thread safe HashMap

<https://leetcode.com/discuss/interview-experience/369839/adobe-sse-interview-experience-waiting>

[**https://www.geeksforgeeks.org/introduction-of-er-model/**](https://www.geeksforgeeks.org/introduction-of-er-model/)

**DATA MODEL:**

[**https://www.gohired.in/2020/02/03/system-design-designing-a-lld-for-hotel-booking/**](https://www.gohired.in/2020/02/03/system-design-designing-a-lld-for-hotel-booking/)

[**https://www.gohired.in/2020/02/03/system-design-designing-a-lld-for-hotel-booking/**](https://www.gohired.in/2020/02/03/system-design-designing-a-lld-for-hotel-booking/)

**>>>>>>>>>>>>>**

**Microservices pattern:**

[**https://www.youtube.com/watch?v=9QTsXLB6Al8&t=2551s**](https://www.youtube.com/watch?v=9QTsXLB6Al8&t=2551s)

**CIRCUIT BREAKE:**

[**https://levelup.gitconnected.com/how-to-implement-a-circuit-breaker-in-microservice-with-alibaba-sentinel-d645f69b20ef**](https://levelup.gitconnected.com/how-to-implement-a-circuit-breaker-in-microservice-with-alibaba-sentinel-d645f69b20ef)

**Scaling backend:**

[**https://www.linkedin.com/pulse/how-scale-backend-pradeep-kumar/**](https://www.linkedin.com/pulse/how-scale-backend-pradeep-kumar/)

**MICROSERVICES RESILIENCE PATTERN**

**1. using retry pattern**

[**https://reflectoring.io/retry-with-resilience4j/**](https://reflectoring.io/retry-with-resilience4j/)

**Graph database:**

[**https://www.youtube.com/watch?v=Dmxq90gJ2xE**](https://www.youtube.com/watch?v=Dmxq90gJ2xE)

**Neo4j**

[**https://www.youtube.com/watch?v=zNUzkvptw8E&list=PLqfPEK2RTgChcOZ6qHgSfwiBPCz2Bzdjh**](https://www.youtube.com/watch?v=zNUzkvptw8E&list=PLqfPEK2RTgChcOZ6qHgSfwiBPCz2Bzdjh)

**CoreJava:**

[**https://github.com/RameshMF/core-java-developers-guide**](https://github.com/RameshMF/core-java-developers-guide)

**CORE JAVA NOTES:**

**1.**

**Memory/heap model in java8**

**2.**

**Compare objects without overriding object and hashCode : reference comparision**

**3.**

**Serialization**

**4.**

**Object Class APIs**

**Finalize??**

**5.**

**Producer/Consumer**

**6. hashCode vs equals??**

**7.**

**Generics**

**Normalization:**

[**https://aksakalli.github.io/2012/03/12/database-normalization-and-normal-forms-with-an-example.html**](https://aksakalli.github.io/2012/03/12/database-normalization-and-normal-forms-with-an-example.html)

**8.**

**Classes type in Java; static and non-static**

**9.**

**Reflection…**

**MultiThreading:**

[**https://github.com/callicoder/java-concurrency-examples/tree/master/java-thread-and-runnable-examples/src**](https://github.com/callicoder/java-concurrency-examples/tree/master/java-thread-and-runnable-examples/src)

[**https://github.com/LeonardoZ/java-concurrency-patterns**](https://github.com/LeonardoZ/java-concurrency-patterns)

**(All Concurrency examples……….)**

**TO DO:**

countdownLatch???

CompletableFuture????

ForkJoinFramework

**MULESOFT:??**

**Information Infosphere**

**Java Questions:**

**Why exception.printStackTrace considered bad practice???**

**Neo4j: Graph DB in java**

**Resume**

**Implementation in product**

**Adapter Pattern: from Wikipedia**

[**https://github.com/fedeoliv/microservices-transactions**](https://github.com/fedeoliv/microservices-transactions)

**RabbitMQ and Choreograogy SAGA imple using springBoot**

**Implement wrt RabbitMQ**

**OAuth 2( read from medium.com)**

**DataModel:**

**Type: conceptual, logical,physical data model**

**Corellated vs Uncorellated queries**

**CDN:**

[**https://medium.com/pixboost/cdn-explained-why-when-and-how-to-use-it-for-your-website-7d360a93cc04**](https://medium.com/pixboost/cdn-explained-why-when-and-how-to-use-it-for-your-website-7d360a93cc04)

**How do SSL prevents DoS Attack???**

**Symettric Vs Asymmetric Encryption?? How is Symmetric safe**

[**https://medium.com/@namangupta01/how-ssl-works-23d8e5ed0cfa**](https://medium.com/@namangupta01/how-ssl-works-23d8e5ed0cfa)

**OAuth:**

[**https://medium.com/swlh/understanding-oauth-2-0-dc7ef422d915**](https://medium.com/swlh/understanding-oauth-2-0-dc7ef422d915)

**TODO**

## **Multidimensional Schema**

**Designed for analytical purposes:**

**Types:**

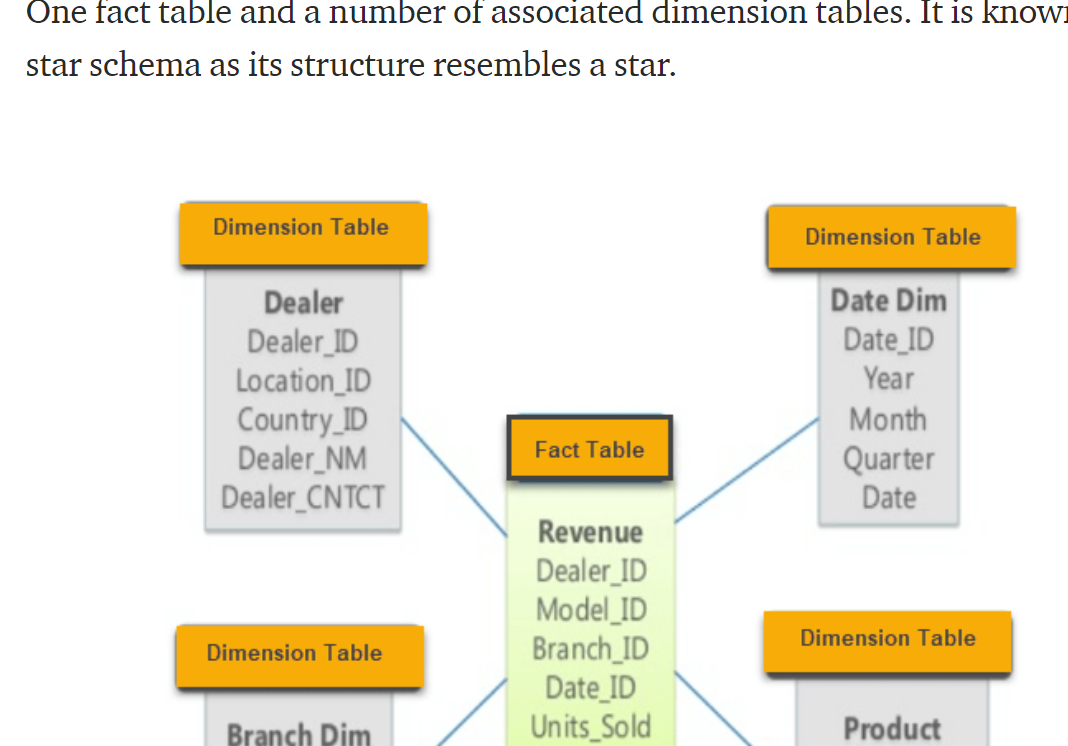
**1.**

**Star Schema**

**2. Snowflake Schema**

**Snowflake is extension of star schema wherein the Dimensions tables are normalized**

**Star Schema**



**Java:**

**MultiThreading**

[**http://tutorials.jenkov.com/java-concurrency/index.html**](http://tutorials.jenkov.com/java-concurrency/index.html)

**Concurreny questions:**

[**https://stackoverflow.com/questions/51686824/handle-concurrency-java-level-vs-database-jpa-level**](https://stackoverflow.com/questions/51686824/handle-concurrency-java-level-vs-database-jpa-level)

**Pessimistic**

**Overhead on the DB**

**Optimistic locking**

**Overhead on the Clinet??? How**

When traffic levels rise, optimistic locking is a better choice.

<https://stackoverflow.com/questions/129329/optimistic-vs-pessimistic-locking>

High concurrency scenarios are good candidates for pessimistic locking.

Optimistic locking requires the client application to refresh its data when a conflict arises. This refresh increases the number of database roundtrips and thus the load at the database server

[**https://medium.com/javarevisited/concurrency-and-locking-on-databases-daa14e2aa68d**](https://medium.com/javarevisited/concurrency-and-locking-on-databases-daa14e2aa68d)

**Data Consistency:**

[**https://medium.com/garantibbva-teknoloji/data-consistency-in-microservices-architecture-5c67e0f65256**](https://medium.com/garantibbva-teknoloji/data-consistency-in-microservices-architecture-5c67e0f65256)

**Moving from monolithic to microservices architectures**

transaction management and data consistency/integrity.

Business logic is spanned to the multiple local transactions

* **Distributed transactions**
* **Eventual consistency**

**How to implement the 2Phase Commits? In distributed Architecture**

**Always Deal with +ves and -ves**

**Transaction CO-ordinator in 2Phase commit?**

**2Phase commit:**

the **two-phase commit protocol** (**2PC**) is a type of [atomic commitment protocol](https://en.wikipedia.org/wiki/Atomic_commit) (ACP)

Eventual consistency uses the **BASE** database model.

***SAGA****is a common pattern that operates the eventual consistency model*

**Type of Saga Pattern: and implementation?**

Event Store is a message broker acting as an event database

**Avoid**using **distributed transactions** across microservices if possible. Why?

<https://medium.com/garantibbva-teknoloji/data-consistency-in-microservices-architecture-5c67e0f65256>

**API Gateway:**

**InMemoryCache blog**

**Transactionsl Vs non-Transactional Resources**

**In Memory Database:**

**How much data can be kept in inmemory database??**

**Ex: memcached**

The trick here is that you still keep everything in memory, but additionally you persist each operation on disk in a transaction log

what our in-memory database does is it floods the disk with transactions as fast as 100 Mbytes per second.

So, magnetic disks are pretty fast when you use them sequentially. On the other hand, they’re utterly slow when you use them randomly

disk-based databases require specific data structures to avoid a full scan of a transaction log in order to read from a dataset fast. One such data structure of the kind is a [B/B+ tree](https://en.wikipedia.org/wiki/B%2B_tree). The flip side of this data structure is that you should change a B/B+ tree on each change operation, which could constitute random workload on a disk. While improving the performance for read operations, B/B+ trees are degrading it for write operations.

LSM Tree as alternative to B/B+ Tree

In [computer science](https://en.wikipedia.org/wiki/Computer_science), the **log-structured merge-tree** (or LSM tree) is a [data structure](https://en.wikipedia.org/wiki/Data_structure) with performance characteristics that make it attractive for providing [indexed](https://en.wikipedia.org/wiki/Database_index) access to files with high insert volume, such as [transactional log data](https://en.wikipedia.org/wiki/Transaction_log). LSM trees, like other [search trees](https://en.wikipedia.org/wiki/Search_tree), maintain key-value pairs. LSM trees maintain data in two or more separate structures, each of which is optimized for its respective underlying storage medium; data is synchronized between the two structures efficiently, in batches.

The last but not least topic that I want to partially cover here is snapshotting. Snapshotting is the way transaction logs are compacted. A snapshot of a database state is a copy of the whole dataset. A snapshot and latest transaction logs are enough to recover your database state. So, having a snapshot, you can delete all the outdated transaction logs that don’t have any new information on top of the snapshot

Why would we need to compact logs? Because the more transaction logs, the longer the recovery time for a database

**Data Latency:**

<https://luanjunyi.medium.com/trade-offs-in-distributed-system-part-1-bb8677a05ffb>

Serializability means each concurrent operation(transaction) on the system must be able to execute as if it’s the only operation. Put it another way, when a set of operations(e.x. <O1, O2, O3, O4, O5>) are processed in parallel, the end result must be equal to some sequential(non-parallel) execution of the set.

SLA-Availability means the percentage of valid requests can be served(for example, Cloud Spanner offers 99.999% availability)

Latency:

Any user facing read requests should have end to end latency less than 1 second

Linearizability is defined in a world where all operations(read/write on a register) are atomic. Operations happen instantaneously and the time of happening is a unique timestamp in the real world. No two operations share the same timestamp(concurrent operations)

QPS:

Throughput(QPS) is measured as the number of requests processed per second. Latency is measured by seconds of request processing time.

**Distributed Transactions in MicroServies:**

<https://medium.com/swlh/handling-transactions-in-the-microservice-world-c77b275813e0>

Two Phase commit :

Advantages:

Atomic nd Asyncronous, read0write Isolation

Implementation of SAGA Choreography:

<https://www.vinsguru.com/choreography-saga-pattern-with-spring-boot/>

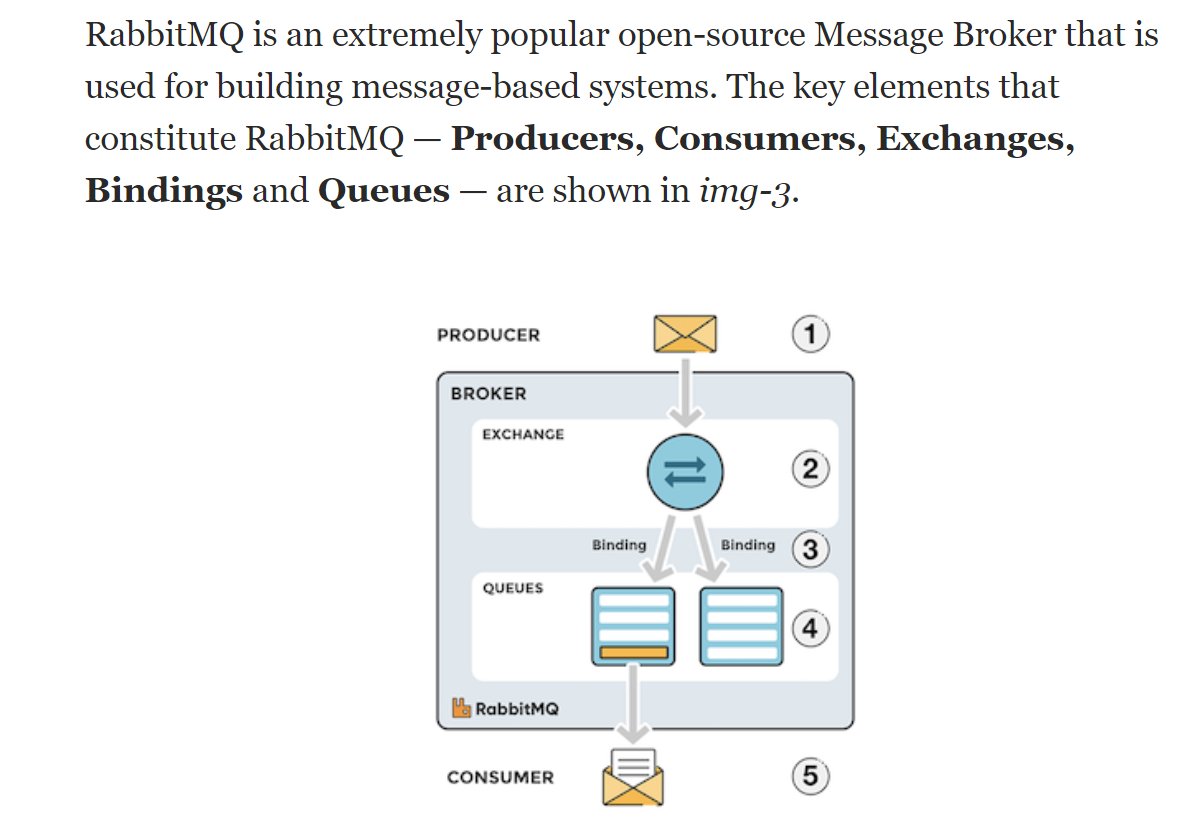
Each business transaction which spans multiple Microservices are split into Microservice specific local transactions and they are executed in a sequence to complete the business workflow

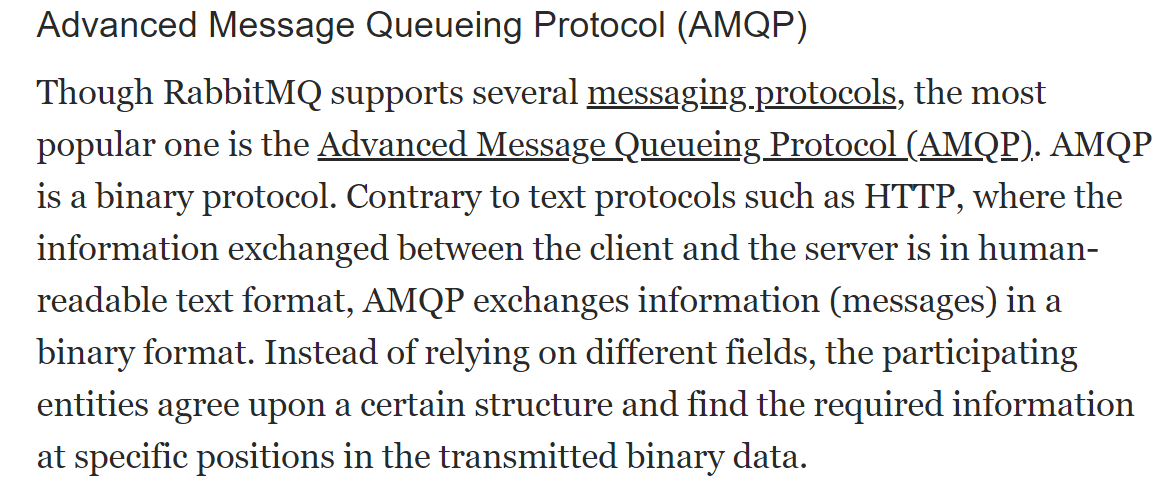
Event Sourcing:

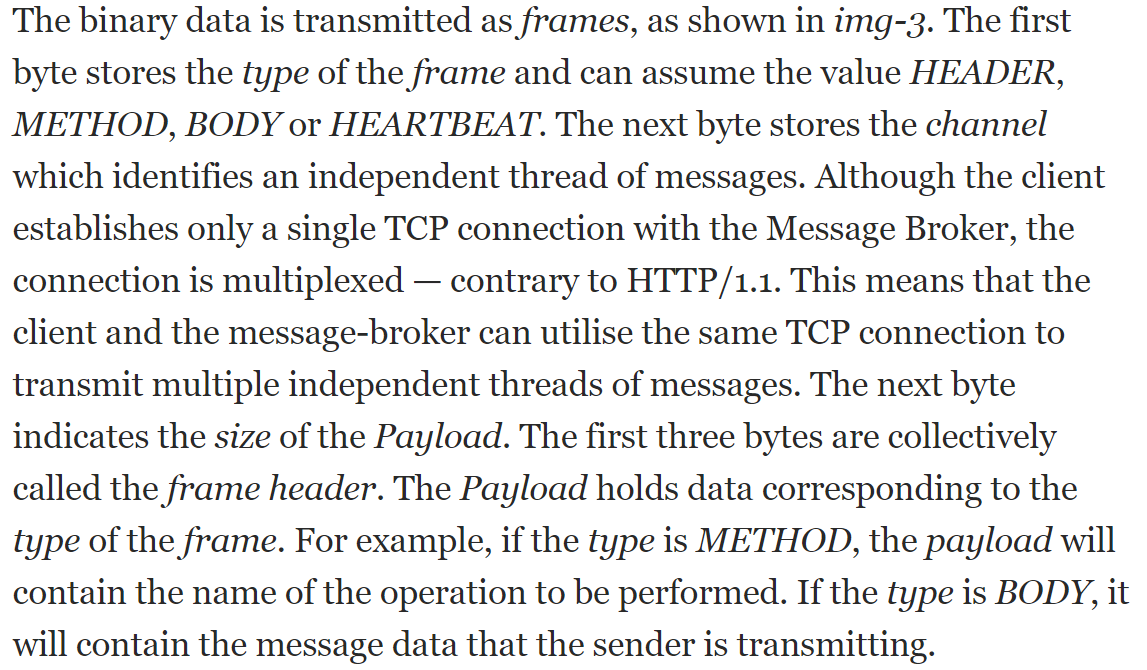
In this approach every change to the state of an application is captured as an ***event***. This event is stored in the database /event store (for tracking purposes) and is also published in the event-bus for other parties to consume

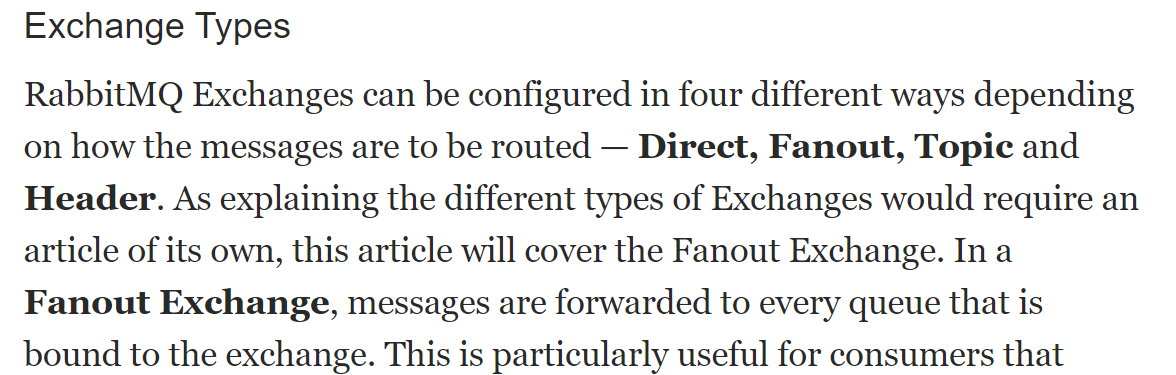
The event object will be named in the past tense always as it already happened.

**Kafka Vs RabbitMQ**









**to do:**

**Java8 static import**

**Create the annotations**

**Sleep Vs wait**

**How reentrant fairness policy works???**

**Mutex vs semaphore vs monitor**

**Synchronized block vs synchronized method**

**DeadLock by wait-notify and by Thread.sleep**

**ThreadPool**

**BlockingQueue**

**DeadLock using wait and notify**

**Adv. Executor Service**

**How to make ReEntrant Lock to re-enter and is it sync wether**

**Object lock is on instance or class level**

**Java heap**

**Finalize**

**FixedThreadPool vs cachedPool**

**Semaphore vs mutex**

**executorService; awaitTermination vs fork Join**

**scheduledThreadPool**

**how to stop the thread**

newScheduledThreadPool

**RabbitMQ for communication between microservices:**

[**https://springframework.guru/spring-boot-messaging-with-rabbitmq/**](https://springframework.guru/spring-boot-messaging-with-rabbitmq/) **in progress…**

[**https://www.cloudamqp.com/blog/why-use-rabbitmq-in-a-microservice-architecture.html**](https://www.cloudamqp.com/blog/why-use-rabbitmq-in-a-microservice-architecture.html)

[**https://github.com/jonashackt/spring-rabbitmq-messaging-microservices**](https://github.com/jonashackt/spring-rabbitmq-messaging-microservices)

**monocots1.fyre.ibm.com**

**for db2:**

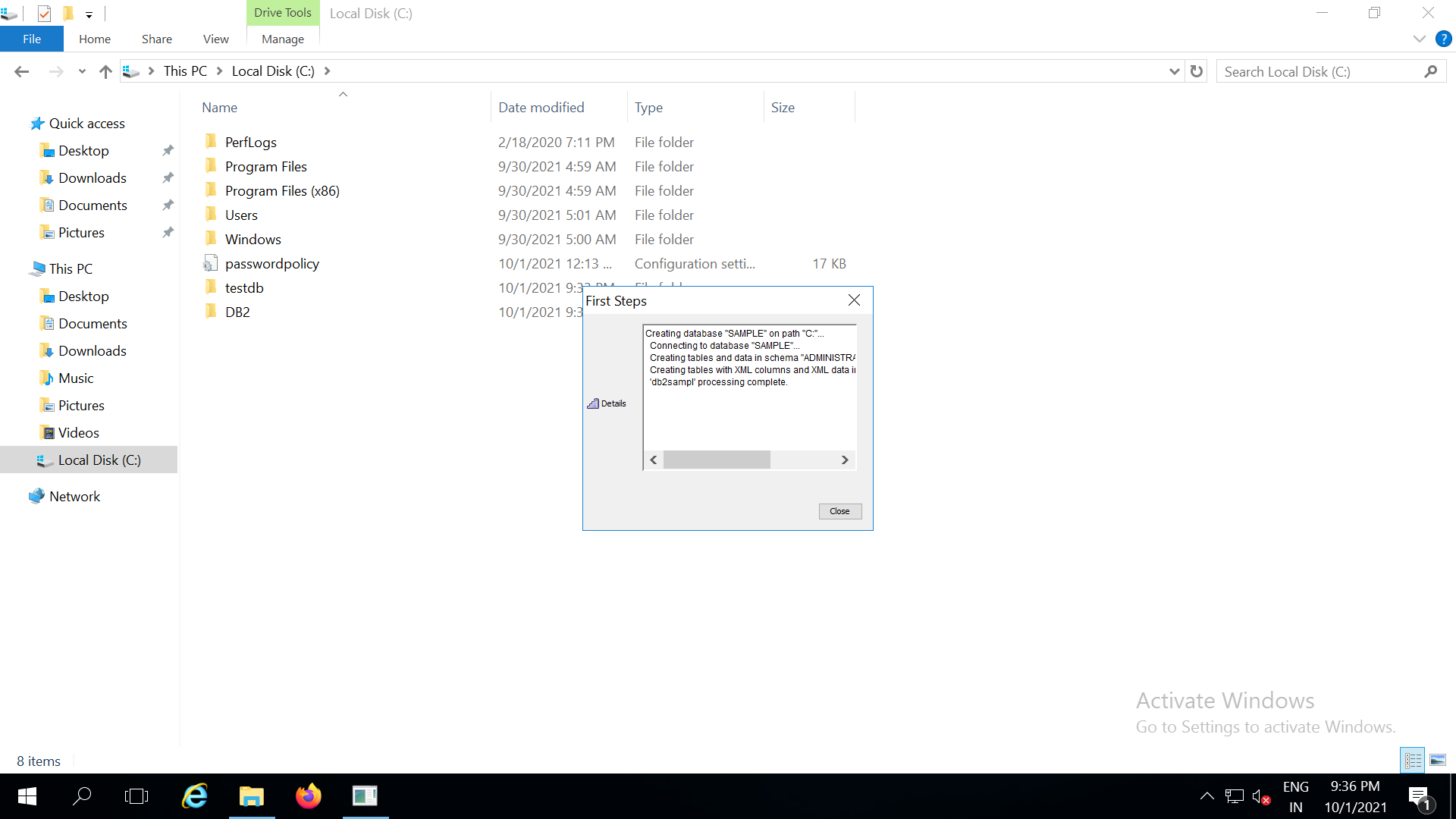
**db2admin/IBM\_0123**

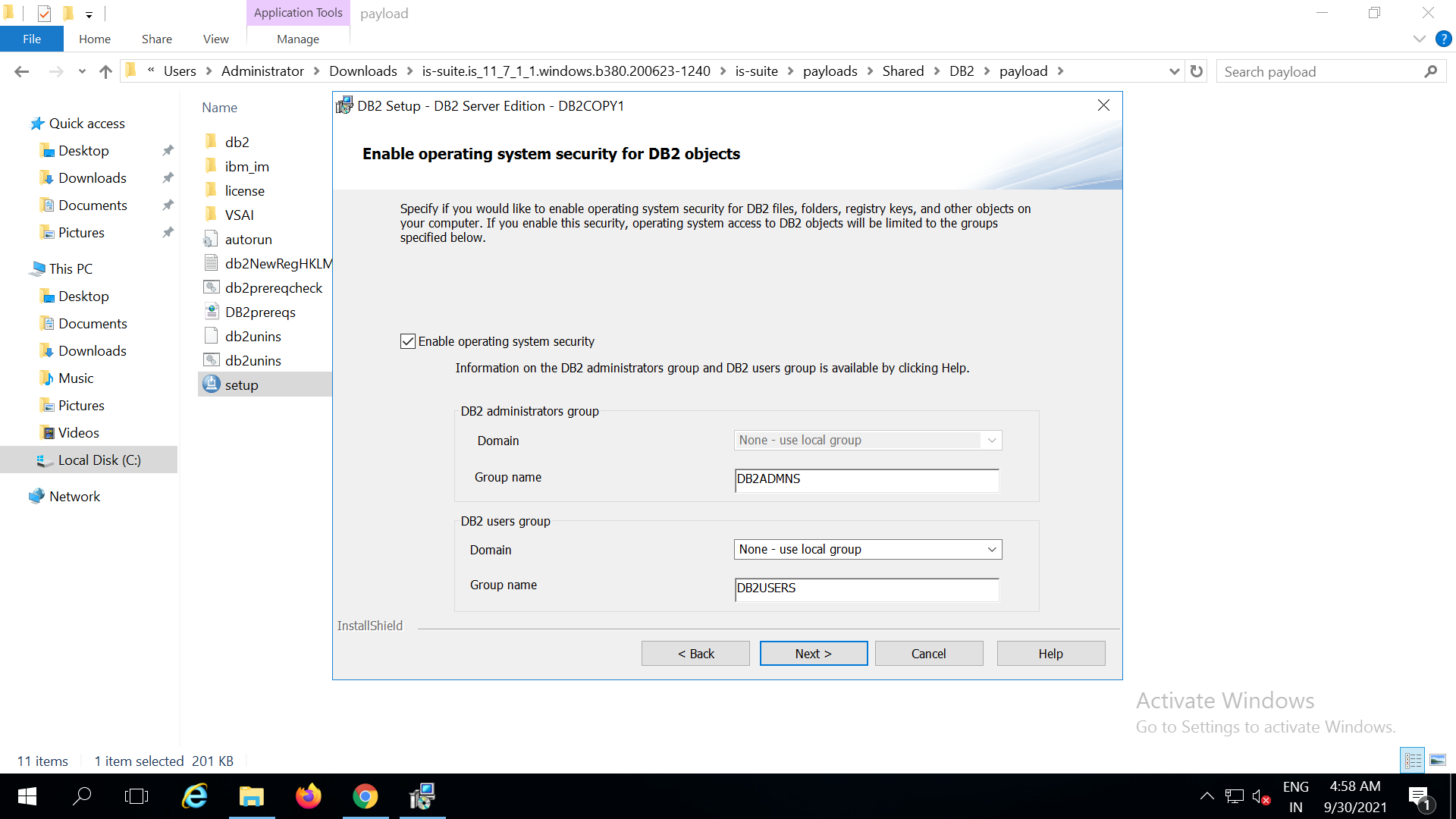
**TO CONNECT TO DB:**

**Db2admin/db2admin**

**DB is: SAMPLE**

**Schema: ADMINISTRATOR**





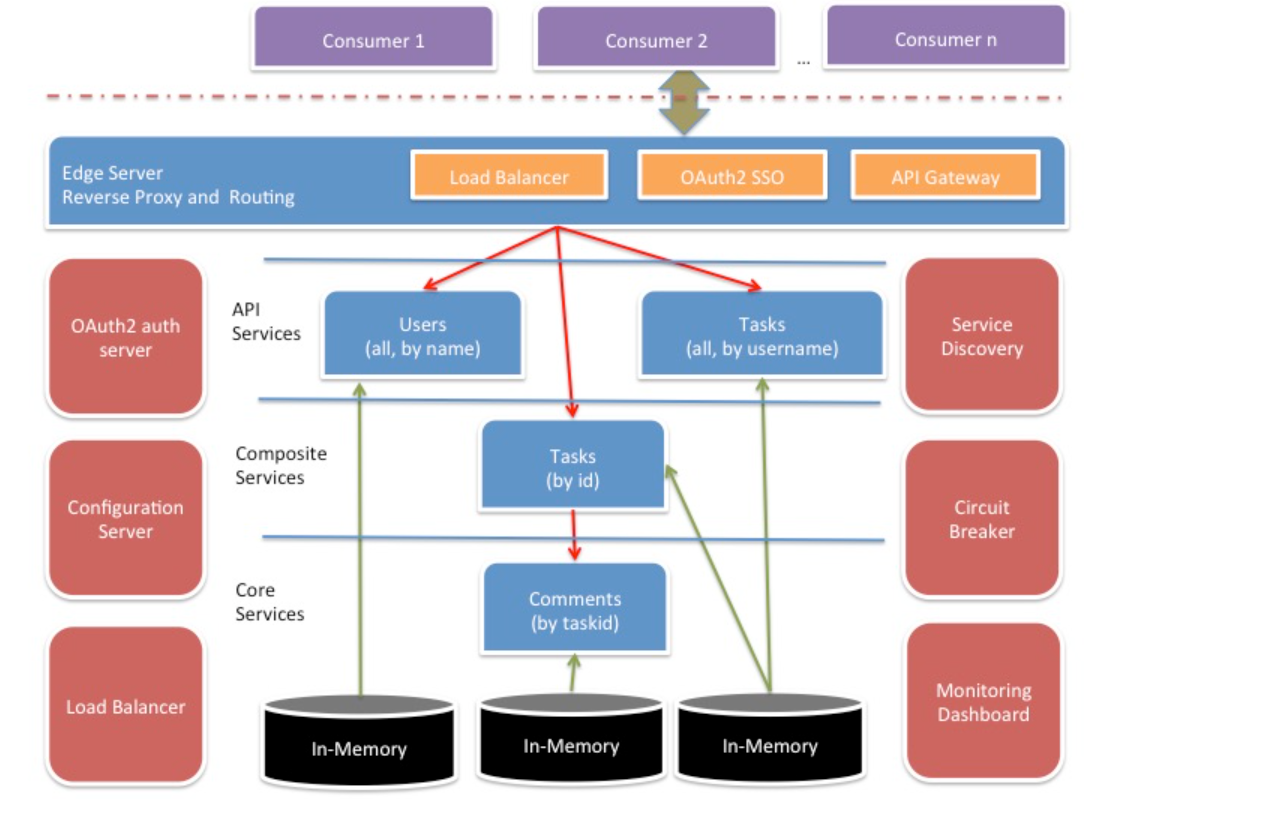
**SQL queries:**

[**https://www.youtube.com/watch?v=zvuWEysWoLE**](https://www.youtube.com/watch?v=zvuWEysWoLE)

**Syncronization at java application level Vs at Database level**

**microservices:**

1. User service
2. OAuthService
3. LoadBalancerService



**Monolithic to microservice migration**

[**https://medium.com/design-microservices-architecture-with-patterns/monolithic-to-microservices-architecture-with-patterns-best-practices-a768272797b2**](https://medium.com/design-microservices-architecture-with-patterns/monolithic-to-microservices-architecture-with-patterns-best-practices-a768272797b2)

**DB2**

**Spring rest : In process:**

[**https://www.youtube.com/watch?v=th3uIP7D8nk**](https://www.youtube.com/watch?v=th3uIP7D8nk)

**Spring registery and discovery:**

[**https://www.youtube.com/watch?v=BnknNTN8icw**](https://www.youtube.com/watch?v=BnknNTN8icw)

**Multi tenant:**

[**https://www.youtube.com/watch?v=eUEiysJX-L0**](https://www.youtube.com/watch?v=eUEiysJX-L0)

[**https://www.youtube.com/watch?v=Kb6qPHAxURU&list=PLm\_DiL9tiiZEf1U1SCO\_xX8ThzYSMCbo7**](https://www.youtube.com/watch?v=Kb6qPHAxURU&list=PLm_DiL9tiiZEf1U1SCO_xX8ThzYSMCbo7)

**Low level deign:**

[**https://workat.tech/machine-coding/practice/design-distributed-queue-cuudq0sk0v14**](https://workat.tech/machine-coding/practice/design-distributed-queue-cuudq0sk0v14)

**Task Scheduler:**

[**https://leetcode.com/discuss/interview-question/system-design/344524/amazon-design-a-jobtask-scheduler**](https://leetcode.com/discuss/interview-question/system-design/344524/amazon-design-a-jobtask-scheduler)

Design memory management subsystem

1. **Microservices**

**Spring Boot:**

[**https://www.youtube.com/watch?v=BnknNTN8icw**](https://www.youtube.com/watch?v=BnknNTN8icw)

[**https://www.youtube.com/watch?v=th3uIP7D8nk**](https://www.youtube.com/watch?v=th3uIP7D8nk)

[**https://github.com/anilallewar/microservices-basics-spring-boot**](https://github.com/anilallewar/microservices-basics-spring-boot)

**Spring REST:**

[**https://github.com/RameshMF/spring-boot-tutorial-course/blob/main/student-management-system/src/main/java/net/javaguides/sms/StudentManagementSystemApplication.java**](https://github.com/RameshMF/spring-boot-tutorial-course/blob/main/student-management-system/src/main/java/net/javaguides/sms/StudentManagementSystemApplication.java)

**1.**

[**https://github.com/Microservice-API-Patterns/LakesideMutual**](https://github.com/Microservice-API-Patterns/LakesideMutual)

**2.**

[**https://github.com/wuyichen24/microservices-patterns**](https://github.com/wuyichen24/microservices-patterns)

**3.**

[**https://github.com/rodrigorodrigues/microservices-design-patterns#microservice-diagram**](https://github.com/rodrigorodrigues/microservices-design-patterns#microservice-diagram)

1. **Sping zuul api gateway:**

[**https://howtodoinjava.com/spring-cloud/spring-cloud-api-gateway-zuul/**](https://howtodoinjava.com/spring-cloud/spring-cloud-api-gateway-zuul/)

**Compiled list:**

[**https://leetcode.com/discuss/interview-question/971009/List-of-2020-interview-question-for-Google**](https://leetcode.com/discuss/interview-question/971009/List-of-2020-interview-question-for-Google)

**leetcode:**

[**https://leetcode.com/study-plan/dynamic-programming/?progress=xg1mvwe**](https://leetcode.com/study-plan/dynamic-programming/?progress=xg1mvwe)

**ARRAY:**

<https://leetcode.com/problems/count-submatrices-with-all-ones/discuss/720265/Java-Detailed-Explanation-From-O(MNM)-to-O(MN)-by-using-Stack>

**DP**

https://leetcode.com/problems/partition-equal-subset-sum/discuss/90592/01-knapsack-detailed-explanationFaceBook Final List

<https://leetcode.com/problems/interleaving-string/discuss/32078/DP-Solution-in-Java>

<https://leetcode.com/problems/minimum-insertion-steps-to-make-a-string-palindrome/discuss/470706/JavaC%2B%2BPython-Longest-Common-Sequence>

<https://leetcode.com/problems/validate-stack-sequences/>

<https://leetcode.com/problems/tiling-a-rectangle-with-the-fewest-squares/discuss/414979/Java-back-tracking-solution>

<https://cheonhyangzhang.gitbooks.io/leetcode-solutions/content/solutions-501-550/527-word-abbreviation.html>

<https://leetcode.com/problems/stone-game-ii/>

**2 Pointers:**

<https://leetcode.com/problems/largest-rectangle-in-histogram/discuss/28902/5ms-O(n)-Java-solution-explained-(beats-96)>

<https://leetcode.com/problems/spiral-matrix/>

<https://leetcode.com/problems/maximal-square/discuss/61876/Accepted-clean-Java-DP-solution>

(1 D solution)

<https://just4once.gitbooks.io/leetcode-notes/content/leetcode/two-pointers/723-candy-crush.html>

<https://github.com/Seanforfun/Algorithm-and-Leetcode/blob/master/leetcode/253.%20Meeting%20Rooms%20II.md>

**DFS:**

<https://leetcode.com/problems/maximum-width-of-binary-tree/discuss/106654/JavaC%2B%2B-Very-simple-dfs-solution>

<https://leetcode.com/problems/word-search-ii/discuss/59784/My-simple-and-clean-Java-code-using-DFS-and-Trie>

<https://leetcode.com/problems/path-sum-ii/discuss/36698/Another-accepted-Java-solution>

<https://leetcode.com/problems/convert-sorted-list-to-binary-search-tree/discuss/35476/Share-my-JAVA-solution-1ms-very-short-and-concise>.

**BFS:**

<https://leetcode.com/problems/all-nodes-distance-k-in-binary-tree/discuss/143798/1ms-beat-100-simple-Java-dfs-with(without)-hashmap-including-explanation>

(Second solution in leetcode discuss)

<https://tenderleo.gitbooks.io/leetcode-solutions-/content/GoogleMedium/286.html>

<https://massivealgorithms.blogspot.com/2015/12/leetcode-317-shortest-distance-from-all.html>

3D ARRAY:

<https://leetcode.com/problems/shortest-path-in-a-grid-with-obstacles-elimination/discuss/451796/Java-Straightforward-BFS-O(MNK)-time-or-O(MNK)-space>

**SLIDING WINDOW:**

<https://leetcode.com/problems/sliding-window-maximum/>

must to do:

<https://leetcode.com/problems/count-unique-characters-of-all-substrings-of-a-given-string/>

Blind 75:

<https://leetcode.com/discuss/general-discussion/460599/blind-75-leetcode-questions>

**Binary Search:**

<https://wentao-shao.gitbook.io/leetcode/binary-search/1062.longest-repeating-substring>

String:

<https://leetcode.com/problems/shortest-palindrome/>

<https://leetcode.com/problems/shortest-subarray-with-sum-at-least-k/discuss/143726/C%2B%2BJavaPython-O(N)-Using-Deque>

<https://leetcode.com/problems/set-matrix-zeroes/>

<https://leetcode.com/problems/word-ladder/(https://www.youtube.com/watch?v=M9cVl4d0v04)by> Nick White

**BINARY SEARCH:**

<https://leetcode.com/problems/search-in-rotated-sorted-array/>

(WITH AND WITHOUT DUPLICATES)

========

1. <https://leetcode.com/problems/find-all-anagrams-in-a-string/submissions/>
2. <https://leetcode.com/submissions/detail/538220628/>
3. <https://leetcode.com/problems/number-of-connected-components-in-an-undirected-graph/>
4. <https://leetcode.com/problems/add-binary/discuss/24488/Short-AC-solution-in-Java-with-explanation>
5. <https://leetcode.com/problems/course-schedule/discuss/162743/JavaC%2B%2BPython-BFS-Topological-Sorting-O(N-%2B-E)>
6. <https://leetcode.com/problems/smallest-subtree-with-all-the-deepest-nodes/>
7. <https://leetcode.com/problems/serialize-and-deserialize-binary-tree/>
8. <https://leetcode.com/problems/interval-list-intersections/>
9. <https://leetcode.com/problems/task-scheduler/>
10. <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/>
11. <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iv/>
12. <https://leetcode.com/problems/group-shifted-strings/>
13. <https://leetcode.com/problems/search-in-rotated-sorted-array-ii/>
14. <https://leetcode.com/problems/binary-tree-maximum-path-sum/submissions/>
15. <https://leetcode.com/problems/smallest-subtree-with-all-the-deepest-nodes/discuss/146808/C%2B%2BJavaPython-One-Pass>
16. <https://leetcode.com/problems/insert-interval/submissions/>
17. <https://leetcode.com/problems/count-unique-characters-of-all-substrings-of-a-given-string/discuss/128952/C%2B%2BJavaPython-One-pass-O(N)>
18. <https://leetcode.com/problems/subarray-product-less-than-k/discuss/108861/JavaC%2B%2B-Clean-Code-with-Explanation>

(How the count of subarray is taken)

<https://leetcode.com/problems/longest-increasing-subsequence/discuss/74824/JavaPython-Binary-search-O(nlogn)-time-with-explanation>

(Hows is done using binary Olog(n))

1. <https://leetcode.com/problems/combination-sum-iii/>
2. <https://leetcode.com/problems/partition-to-k-equal-sum-subsets/discuss/180014/Backtracking-Thinking-Process>
3. <https://leetcode.com/problems/binary-tree-maximum-path-sum/submissions/>
4. <https://leetcode.com/problems/expression-add-operators/>
5. <https://leetcode.com/problems/random-pick-with-weight/>
6. <https://leetcode.com/problems/integer-to-english-words/discuss/70625/My-clean-Java-solution-very-easy-to-understand>
7. <https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/discuss/231148/Java-TreeMap-Solution>
8. <https://leetcode.com/problems/interval-list-intersections/>
9. <https://leetcode.com/problems/interval-list-intersections/discuss/231122/Java-two-pointers-O(m-%2B-n)>
10. <https://leetcode.com/problems/minimum-add-to-make-parentheses-valid/discuss/181132/C%2B%2BJavaPython-Straight-Forward-One-Pass>
11. <https://leetcode.com/problems/number-of-subarrays-with-bounded-maximum/submissions/>
12. <https://leetcode.com/problems/binary-search-tree-to-greater-sum-tree>
13. <https://leetcode.com/problems/sliding-window-median/discuss/96346/Java-using-two-Tree-Sets-O(n-logk)>
14. <https://leetcode.com/problems/cherry-pickup-ii/>
15. <https://leetcode.com/problems/stone-game-ii/>
16. <https://leetcode.com/problems/time-needed-to-inform-all-employees/discuss/532560/JavaC%2B%2BPython-DFS>
17. <https://leetcode.com/problems/flip-equivalent-binary-trees/discuss/200514/JavaPython-3-DFS-3-liners-and-BFS-with-explanation-time-and-space%3A-O(n)>.
18. <https://leetcode.com/problems/shortest-subarray-with-sum-at-least-k/discuss/143726/C%2B%2BJavaPython-O(N)-Using-Deque>
19. <https://leetcode.com/problems/24-game/>
20. <https://leetcode.com/problems/critical-connections-in-a-network>
21. <https://leetcode.com/problems/my-calendar-i/submissions/>
22. <https://leetcode.com/problems/my-calendar-i/discuss/109462/Java-8-liner-TreeMap> ; By using Binary Search???
23. <https://leetcode.com/problems/minimum-window-subsequence/>
24. <https://leetcode.com/problems/split-array-largest-sum/>
25. <https://leetcode.com/problems/critical-connections-in-a-network/discuss/382638/DFS-detailed-explanation-O(orEor)-solution>
26. <https://leetcode.com/problems/concatenated-words/discuss/541520/Java-DFS-%2B-Memoization-Clean-code(Good> example of DFS)
27. <https://leetcode.com/problems/word-break-ii/discuss/44167/My-concise-JAVA-solution-based-on-memorized-DFS>
28. <https://leetcode.com/problems/word-ladder-ii/>
29. https://leetcode.com/problems/palindrome-partitioning-iii/discuss/498677/Step-by-Step-solution-DP-(Java)

<https://leetcode.com/problems/count-of-smaller-numbers-after-self/discuss/445769/merge-sort-CLEAR-simple-EXPLANATION-with-EXAMPLES-O(n-lg-n)>

<https://leetcode.com/problems/powx-n/>

<https://leetcode.com/problems/largest-rectangle-in-histogram/>

<https://leetcode.com/problems/remove-duplicates-from-sorted-array/>

<https://leetcode.com/problems/maximal-rectangle/discuss/29054/Share-my-DP-solution>

<https://leetcode.com/problems/longest-string-chain/>

<https://leetcode.com/problems/delete-nodes-and-return-forest/(Good> recursion)

<https://leetcode.com/problems/shortest-path-in-a-grid-with-obstacles-elimination/discuss/451796/Java-Straightforward-BFS-O(MNK)-time-or-O(MNK)-space>

**PHASE 2:**

**SOLVE BY TOPIC:**

[**https://github.com/neerazz/FAANG/blob/master/Algorithms/Top\_LeetCode\_Questions\_By\_Topic.md**](https://github.com/neerazz/FAANG/blob/master/Algorithms/Top_LeetCode_Questions_By_Topic.md)

**Good Stack probles:**

[**https://leetcode.com/problems/minimum-cost-tree-from-leaf-values/discuss/339959/One-Pass-O(N)-Time-and-Space**](https://leetcode.com/problems/minimum-cost-tree-from-leaf-values/discuss/339959/One-Pass-O(N)-Time-and-Space)

[**https://leetcode.com/problems/minimum-cost-tree-from-leaf-values/**](https://leetcode.com/problems/minimum-cost-tree-from-leaf-values/)

[**https://leetcode.com/problems/sum-of-subarray-minimums/discuss/178876/stack-solution-with-very-detailed-explanation-step-by-step**](https://leetcode.com/problems/sum-of-subarray-minimums/discuss/178876/stack-solution-with-very-detailed-explanation-step-by-step)

**(MONOTONE STACK EXAMPLES)**

using STACK:

<https://leetcode.com/problems/remove-duplicate-letters/discuss/76769/Java-solution-using-Stack-with-comments>

<https://leetcode.com/problems/sliding-window-maximum/discuss/65884/Java-O(n)-solution-using-deque-with-explanation>

<https://leetcode.com/problems/maximum-product-subarray/discuss/183483/JavaC%2B%2BPython-it-can-be-more-simple>

<https://leetcode.com/problems/minimum-path-sum/discuss/23477/DP-Solution-Linear-space>( How is it done in O(n))

<https://leetcode.com/problems/longest-valid-parentheses/discuss/14278/Two-Java-solutions-with-explanation.-Stack-and-DP.-Short-and-easy-to-understand>.

<https://leetcode.com/problems/word-break-ii/discuss/44167/My-concise-JAVA-solution-based-on-memorized-DFS>

<https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/discuss/135704/Detail-explanation-of-DP-solution>

<https://leetcode.com/problems/decode-ways/discuss/30451/Evolve-from-recursion-to-dp>

GOOGLE

<https://leetcode.com/discuss/general-discussion/458695/dynamic-programming-patterns>

<https://leetcode.com/problems/minimum-number-of-refueling-stops/discuss/149839/DP-O(N2)-and-Priority-Queue-O(NlogN)>

(DP vs BFS)

<https://leetcode.com/discuss/general-discussion/458695/dynamic-programming-patterns>

<https://leetcode.com/problems/unique-paths-ii/discuss/23250/Short-JAVA-solution>

\*\* simply remove j and make array of size==number of columns

(how by 1 D????? DP)

<https://leetcode.com/problems/target-sum/>

https//leetcode.com/submissions/detail/538220628/

>>>>>>>>>>>>>>>>>>>

LOW LEVEL DESIGN:

Web Crawler:

<https://github.com/yasserg/crawler4j>

OpenScale: ICP4d:

<https://www.ibm.com/docs/en/cloud-paks/cp-data/3.5.0?topic=openscale-installing-watson>

<https://www.ibm.com/docs/en/cloud-paks/cp-data/3.5.0?topic=openscale-setting-up-cluster-watson>

TO Do: Does Open shift knowledge needed before setting up OpenScale

<https://www.ibm.com/docs/en/cloud-paks/cp-data/3.5.0?topic=installing-preparing-install-upgrade-services#svc-install-prep__install-node>

<https://www.youtube.com/watch?v=gBI0ApHUFSs>

Install CPD on OpenShift

<https://www.ibm.com/docs/en/cloud-paks/cp-data/3.5.0?topic=iwo-installing-watson-openscale>

Preparing OpenScale DashBoard:

<https://aiopenscale.cloud.ibm.com/aiopenscale/insights?nocache=true&bss_account=24a58d0f17874647ad866620b11b9e32>

**https://github.com/IBM/watson-openscale-samples**

<https://github.com/IBM/watson-openscale-samples/blob/main/Cloud%20Pak%20for%20Data/WML/notebooks/binary/spark/Watson%20OpenScale%20and%20Watson%20ML%20Engine.ipynb>

To get the Demo done for IBm Watson OpenScale:

<https://aiopenscale.cloud.ibm.com/aiopenscale/insights?nocache=true&bss_account=24a58d0f17874647ad866620b11b9e32>

mistakes