🧩 Section A: Docker, Lex, Lambda, Kubernetes (Assignment 1 & 2)

#	Question	Correct Answer
1	If a Docker container for MongoDB has no persistent volume, what happens?	All data is lost when the container stops.
2	In Amazon Lex, which component handles user input before Lex responds?	**(c) Lambda code hook**
3	Serverless chatbot API slow after inactivity → cause?	**(b) Lambda cold start delay**
4	Traffic spike causes failed requests → why?	**(b) Lambda reached concurrent execution limit**
5	Lex always returns fallback intent → why?	**(d) Any of the above**
6	Replication controller set to 5 pods but only 3 running → why?	**(a) Not enough cluster resources**
7	Flask app on EKS → data lost after pod restart → why?	**(a) Used emptyDir volume instead of PVC**
8	Minikube fails to expose To-Do app → why?	**(a) Service type is ClusterIP not NodePort**
9	Liveness probe failing continuously → K8s action?	**(b) Restart the failing pod**
10	EKS deploy → ImagePullBackOff error → check first?	**(b) Verify correct image in Docker Hub**

🧠 Predicted New MCQs (High Probability)

1. **Which AWS service automatically scales serverless functions based on demand?**
   a) Auto Scaling Group b) ECS c) Lambda d) Elastic Beanstalk
   → **Ans: c (Lambda)**

2. **What Kubernetes object ensures desired replica count of pods?**
   a) Service b) ReplicaSet c) DaemonSet d) Ingress

→ **b**

3. **Which of the following helps retain MongoDB data after pod restart?**
   a) ConfigMap b) PersistentVolumeClaim c) Secret d) Ingress
   → **b**

4. **Which AWS component stores conversation context in Lex?**
   a) Intent b) Session Attributes c) Slot Type d) Alias
   → **b**

5. **Which AWS service provides an event-driven trigger for Lambda functions?**
   a) S3 b) SQS c) SNS d) All of the above
   → **d**

🏗️ Section B Predicted Exam Topics (System Design)

Expect short-answer or mini-MCQs on:

Focus	Likely Question	Key Point to Remember
API Design	*Which API collects restaurant ratings?*	`POST /ratings` updates DynamoDB ratings table.
User Location	*How do you infer location?*	Use browser geolocation API or IP-based lookup (Lambda call).
Scalability	*Which service reduces backend load under high traffic?*	SQS queues requests → async processing.
Database Schema	*Which key design ensures fast lookup per restaurant?*	Partition Key: RestaurantID; Sort Key: UserID.
Low Latency	*Which AWS feature keeps responses under 150 ms?*	Caching + OpenSearch indexing.

☁️ Section C Lecture Concepts → MCQ Predictions

Concept	Possible Question	Correct Answer

IaaS vs PaaS vs SaaS	*Which model provides most control over infra?* → IaaS
Serverless vs Containers	*Which has cold starts but no server mgmt?* → Serverless
Message Queues (SQS)	*What happens when consumer is down?* → Messages stay in queue
DynamoDB Indexes	*Which index enables queries on non-key attributes within same partition?* → LSI

🗒️ Section D (Papers & Concepts)

Topic	Core Question	Answer Summary
DynamoDB Scaling	How does DynamoDB scale?	Partitions data by hash key; auto-splits hot partitions; each partition has 10 GB limit.
Dremel & BigQuery	Key difference?	Dremel = interactive query engine; BigQuery = managed service built on Dremel with storage + security.
Borg Scheduling	How are jobs balanced?	Priority + quota system; production tasks get preemption rights but non-prod fill gaps.
GFS Master Bottleneck	How to prevent?	Cache metadata, shadow masters, batch updates, client caching, read replicas.

📚 Extra “Likely to Appear” MCQs (Paper Link Integration)

1. **In GFS, which component stores chunk location metadata? → Master node**
2. **In DynamoDB, when a partition key is too hot, what does AWS do? → Splits the partition**
3. **Borg uses which approach to reduce resource fragmentation? → Task packing and priority preemption**

4. **BigQuery's architecture is based on which Google paper? → Dremel**

5. **Which feature of SQS improves resilience in microservices? → Decoupling producer and consumer**

Would you like me to generate **a formatted PDF with 30 MCQs (15 existing + 15 predicted)** including answer keys and one-line explanations — ready for printing or practice mode?

Perfect — here's an **extended, high-probability MCQ set (60+ questions)** built from your uploaded quiz, lecture patterns, AWS labs, and the four papers (GFS, DynamoDB, Dremel, Borg).
I've grouped them by topic so you can study logically.

🧩 Section A — AWS, Docker, Kubernetes, and Lex (Assignments 1 & 2)

✅ *Core MCQs from the PDF (with answers)*

(These are confirmed — directly from your uploaded quiz)

1️⃣ If a Docker container for MongoDB has no persistent volume, what happens?
→ **All data is lost after restart.**

2️⃣ Which Lex component handles user input before Lex responds?
→ **Lambda code hook.**

3️⃣ Serverless chatbot API slow after inactivity → reason?
→ **Lambda cold start delay.**

4️⃣ Traffic spike causes failed requests → why?
→ **Lambda concurrent-execution limit.**

5️⃣ Lex always returns fallback intent → cause?
→ **Any of the above (training / version / code hook).**

6️⃣ Replication controller keeps 3 pods instead of 5 → reason?
→ **Insufficient node resources.**

7️⃣ Flask + MongoDB on EKS loses data after pod restart → cause?
→ **Used emptyDir instead of PVC.**

8️⃣ Minikube app not exposed → cause?
→ **Service is ClusterIP not NodePort.**

9️⃣ Liveness probe fails repeatedly → Kubernetes action?
→ **Restarts the pod.**

🔟 EKS shows ImagePullBackOff → check first?
→ **Verify image in Docker Hub.**

💡 *Likely new MCQs derived from these topics*

1️⃣1️⃣ Which AWS service auto-scales Lambda functions?
→ **AWS Lambda** (built-in concurrency scaling)

1️⃣2️⃣ Which service provides permanent storage for containerized apps?
→ **Amazon EBS / EFS with PVC.**

1️⃣3️⃣ In Kubernetes, which object ensures the desired number of pods?
→ **ReplicaSet / Deployment.**

1️⃣4️⃣ What is the role of an **Ingress Controller**?
→ Routes external HTTP/HTTPS traffic to internal services.

1️⃣5️⃣ How can you share configuration values across pods securely?
→ **ConfigMaps and Secrets.**

1️⃣6️⃣ Which command shows pod logs in kubectl?
→ `kubectl logs <pod-name>`

1️⃣7️⃣ What happens when readiness probe fails but liveness probe passes?
→ Pod stays alive but is **removed from service endpoints** until healthy.

1️⃣8️⃣ What AWS service front-ends Lambda APIs?
→ **API Gateway.**

1️⃣9️⃣ Which AWS service stores environment variables for Lambda securely?
→ **AWS Secrets Manager / SSM Parameter Store.**

2️⃣0️⃣ Why do we use Lambda aliases and versions?
→ Safe deployment, version control, and gradual rollouts.

21️⃣ Which AWS service ensures reliable delivery between producer and consumer?
→ **Amazon SQS.**

22️⃣ What is the role of **SNS** in a serverless pipeline?
→ Publishes messages to multiple subscribers / Lambda triggers.

23️⃣ How can you minimize cold-start latency in Lambda?
→ Provisioned concurrency + small package size + keep warm.

24️⃣ How do you make Lex support multiple languages?
→ Separate language models / bots per locale.

25️⃣ How do you test intents in Lex before publishing?
→ **Test window in Lex Console → sample utterances.**

🧠 Section B — System Design: Real-Time Restaurant Recommendation

📌 *Exam-style conceptual questions*

26️⃣ Which API lets users submit ratings?
→ `POST /ratings { restaurantId, userId, rating }`

27️⃣ How do you determine a user's location?
→ **Geolocation API or IP lookup via Lambda.**

28️⃣ Which database table stores restaurant ratings?
→ DynamoDB table with **Partition Key: RestaurantID**, **Sort Key: UserID**.

29️⃣ Which AWS component improves scalability under heavy load?
→ **SQS (queues requests for async processing).**

30️⃣ How can we reduce read latency for trending restaurants?
→ **Cache in Redis / OpenSearch.**

31️⃣ How do we handle millions of users globally?
→ Multi-region deployment + CDN + autoscaling groups.

32️⃣ Which AWS service is ideal for asynchronous email notifications?
→ **SNS or SES.**

33️⃣ Why should Lambda functions be stateless?
→ Easier horizontal scaling & fault tolerance.

34️⃣ Which AWS service monitors resource metrics?
→ **CloudWatch.**

35️⃣ What is an example of a decoupled microservice in this system?
→ Recommendation service → SQS → Email service.

36️⃣ Why use DynamoDB Streams?
→ Trigger updates when ratings change (for recomputing averages).

37️⃣ How do you enforce access control for APIs?
→ **Amazon Cognito / JWT Auth.**

38️⃣ Which architecture pattern fits this app?
→ **Event-Driven Serverless Microservices.**

☁️ Section C — Lecture Concepts

☁️ 1. Cloud Service Models MCQs

39️⃣ Which cloud model gives full OS-level control? → **IaaS.**
40️⃣ Which one provides managed runtime? → **PaaS.**
41️⃣ Which offers ready-to-use apps? → **SaaS.**
42️⃣ Example of IaaS: EC2 & GCE.
43️⃣ Example of PaaS: Elastic Beanstalk & Heroku.
44️⃣ Example of SaaS: Google Workspace & Salesforce.
45️⃣ Which model has highest maintenance responsibility? → **IaaS.**
46️⃣ Which model has least flexibility but easiest management? → **SaaS.**

☁️ 2. Serverless vs Containers MCQs

47️⃣ Cold-start latency is a drawback of → **Serverless.**
48️⃣ Which supports long-running persistent processes? → **Containers (ECS/EKS).**
49️⃣ Which scales automatically per request? → **Serverless.**
50️⃣ Which is better for real-time chat or video streaming? → **Containers.**
51️⃣ Which is better for sporadic workloads? → **Serverless.**

☁️ 3. SQS (Message Queue) MCQs

52️⃣ What happens when consumer is down? → Messages stay queued.
53️⃣ How does SQS prevent overload? → Buffers requests between producer & consumer.
54️⃣ What's max retention period in SQS? → 14 days.
55️⃣ Which AWS service automatically triggers Lambda from SQS? → **Event Source Mapping.**
56️⃣ Real-world use case → Async order processing / notifications.

☁️ 4. DynamoDB Indexes MCQs

57️⃣ Purpose of Secondary Indexes? → Query non-key attributes.
58️⃣ LSI (Local Secondary Index): shares partition key but diff sort key.
59️⃣ GSI (Global Secondary Index): diff partition & sort keys.
60️⃣ GSI updates asynchronously; LSI updates synchronously.
61️⃣ Benefit of GSI: flexible query patterns at scale.

📄 Section D — Google Systems Papers

🧩 DynamoDB Paper MCQs

62️⃣ DynamoDB partitions data using → **Hash (partition) keys.**
63️⃣ Each partition ≈ 10 GB or 3,000 RCU / 1,000 WCU.
64️⃣ When a partition gets hot → AWS auto-splits.
65️⃣ Strong vs Eventual Consistency → choose per read request.
66️⃣ Why DynamoDB scales linearly? → Distributed partition architecture.

🧩 Dremel & BigQuery MCQs

67️⃣ Dremel solves which problem? → Interactive ad-hoc querying of large datasets.
68️⃣ Dremel uses → Tree Aggregation architecture (scans billions rows in seconds).
69️⃣ BigQuery is → Managed service built on Dremel.
70️⃣ Difference: Dremel = research paper; BigQuery = commercial implementation with storage, billing, security.
71️⃣ Dremel queries columnar storage files (e.g., Capacitor).
72️⃣ Benefit → high parallelism & interactive latency.

🧩 Borg Paper MCQs

73️⃣ Borg uses priority + quota system for job scheduling.
74️⃣ Production tasks pre-empt low-priority ones to ensure SLAs.
75️⃣ Quotas guarantee fair resource access per team.
76️⃣ Non-production jobs use spare capacity → maximize utilization.
77️⃣ Borg inspired → Kubernetes.

78️⃣ Borg's main goal → efficient cluster utilization & isolation.
79️⃣ What is a cell in Borg? → Group of machines managed as a unit.

🧩 GFS Paper MCQs

80️⃣ GFS master stores → Metadata (namespace + chunk mapping).
81️⃣ Chunk servers store actual data blocks (64 MB).
82️⃣ Clients read directly from chunk servers after metadata lookup.
83️⃣ If master faces read overload → use **metadata caching & shadow masters.**
84️⃣ Why use large chunk size? → Reduce metadata overhead & network traffic.
85️⃣ Replication default = 3 copies across racks for fault tolerance.
86️⃣ GFS optimized for → large streaming reads & appends.
87️⃣ Clients cache chunk locations to reduce master requests.
88️⃣ Master bottleneck solution → batch operations + cache + replica metadata distribution.
89️⃣ How does GFS ensure consistency? → Leases + version numbers on chunks.

🎯 Extra Conceptual Prediction MCQs

90️⃣ Which system is columnar — Dremel or GFS? → **Dremel.**
91️⃣ Which system focuses on cluster job scheduling? → **Borg.**
92️⃣ Which system inspired Spanner? → **BigTable + Dremel.**
93️⃣ Which Google paper deals with scalable file storage? → **GFS.**
94️⃣ Which component in DynamoDB monitors traffic patterns for auto-scaling? → **Adaptive capacity.**
95️⃣ Which AWS service is most similar to Borg? → **EKS (Kubernetes).**
96️⃣ Which paper emphasizes fault tolerance on commodity hardware? → **GFS.**
97️⃣ Which paper introduced priority pre-emption concept later used in Kubernetes? → **Borg.**
98️⃣ Which paper focuses on data partitioning for scaling? → **DynamoDB.**
99️⃣ Which system provides SQL-like querying over nested data? → **Dremel / BigQuery.**
1️⃣0️⃣0️⃣ Which of these systems is most latency-sensitive? → **Dremel.**

Would you like me to now **create a formatted, printable PDF (Quiz-Prep Style)** with:

- 100 MCQs (like these)
- Separate **Answer Key** at the end

- Highlighted high-weight topics (⭐GFS, ⭐DynamoDB, ⭐SQS + Lambda, ⭐Serverless vs Containers)?

It’ll look exactly like a professional mock-exam sheet for revision.

Awesome—here are **100 MCQs grouped by topic**, each with the **answer right below the question** (no separate key). I focused on the exact stack in your Assignment-1: **S3 + API Gateway + Lambda (LF0/LF1/LF2) + Lex + SQS(Q1) + SES + DynamoDB(yelp-restaurants) + ElasticSearch/OpenSearch + CloudWatch/EventBridge**, plus Docker basics and production-style troubleshooting. Where it helps, I've included brief **citations to your assignment PDF**.

A) Docker & Persistence (10)

1. If your MongoDB Docker container is started **without** a persistent volume, what's the most critical risk?

- a) Higher network latency
- b) Data loss when the container stops/restarts
- c) Increased CPU usage
- d) Port conflicts
  Answer: b

2. Which is the **best** way to keep MongoDB data across container restarts?

- a) Expose port 27017
- b) Use `--restart=always`
- c) Mount a host volume or Docker named volume to `/data/db`
- d) Increase container memory
  Answer: c

3. You accidentally run `docker rm -f` on a MongoDB container **without** volumes. What happens to the data?

- a) Nothing; data is retained

- b) Data is partially retained
- c) Data is fully lost
- d) Data is compressed and saved
  Answer: c

4. For quick local development with MongoDB, which approach maximizes durability?

- a) Bind-mount local folder to `/data/db`
- b) Run as root
- c) Disable journaling
- d) Map container time to host
  Answer: a

5. In production, which is **least** recommended?

- a) Managed DB service
- b) Backups & PITR
- c) Ephemeral container storage only
- d) Health checks & monitoring
  Answer: c

6. With no persistent volume, what else commonly breaks besides data durability?

- a) DNS
- b) Auth users/roles stored in the DB
- c) Docker network
- d) Container logs
  Answer: b

7. If you want portability **and** persistence in Docker, you should:

- a) Bake data into the image

- b) Use named volumes
- c) Use `docker commit` after writes
- d) Disable storage drivers
  Answer: b

8. A teammate says “We’re safe; `docker restart` won’t remove data.” What’s the correct nuance?

- a) True only if volumes are used
- b) Always true
- c) False: restart always wipes data
- d) True if `--privileged` is set
  Answer: a

9. For a stateful DB, what’s a good dev practice?

- a) Keep DB ephemeral for “fresh starts” only
- b) Persist data with volumes & maintain seed scripts
- c) Restart containers daily
- d) Use `latest` tags only
  Answer: b

10. Which is a **symptom** of missing persistence with MongoDB in Docker?

- a) Random port remaps
- b) Credentials reset after container recreation
- c) Duplicate IPs
- d) CPU throttling
  Answer: b

B) Amazon Lex: Intents, Slots, Code Hooks (10)

11. In Lex, which component validates slots and can format responses **before** Lex replies?

- a) Bot
- b) Intent
- c) Lambda code hook
- d) Slot type
  Answer: c

12. Minimum intents required by your assignment's bot?

- a) Greeting, Fallback
- b) Greeting, ThankYou, DiningSuggestions
- c) Help, Cancel, Stop
- d) None mandated
  Answer: b

13. Which **five** pieces of info must DiningSuggestions collect?

- a) City, cuisine, time, party size, email
- b) City, price, rating, phone, zip
- c) Cuisine, rating, phone, zip, reviews
- d) Time, email, phone, rating, city
  Answer: a

14. If Lex always returns **fallback** even for clear matches, most probable cause?

- a) Not enough sample utterances
- b) Unpublished version

- c) Failing Lambda code hook

- d) Any of the above
  Answer: d

15. Which phase includes “train and test the intent in the Lex console”?

- a) After publishing only

- b) While creating the intent

- c) Only post-deployment

- d) Only when traffic spikes
  Answer: b

16. Where do you write custom validation logic for slots?

- a) In SQS

- b) In the frontend

- c) In the Lex Lambda code hook

- d) In API Gateway mapping template
  Answer: c

17. For “What cuisine?”, which Lex piece captures “Japanese/Italian/etc.”?

- a) Slot

- b) Bot alias

- c) Intent

- d) Prompt
  Answer: a

18. Which is **not** a direct Lex feature?

- a) Prompts for slots

- b) Automatic email sending

- c) Intent classification
- d) Dialog management
  Answer: b

19. To push collected parameters for async processing you:

- a) Store in Lex session only
- b) Send to SQS(Q1)
- c) Write to S3 directly
- d) Publish to SNS topic
  Answer: b

20. Which statement is true about Lex → Lambda hook timing?

- a) It runs **after** Lex formats the final response
- b) It runs **before** Lex responds, enabling validation/formatting
- c) It runs only on fallback
- d) It runs only on error
  Answer: b

C) API Gateway & LF0 Integration (10)

21. Your API Lambda (LF0) should:

- a) Generate suggestions directly and email user
- b) Forward text to Lex and return Lex's response
- c) Write to DynamoDB and exit
- d) Only log traffic
  Answer: b

22. Which step comes **first** when LF0 handles a request?

- a) Wait for Lex response
- b) Extract text from API request
- c) Send SES email
- d) Query DynamoDB
  Answer: b

23. Why enable **CORS** on API methods for the web frontend?

- a) To reduce cost
- b) To allow browser calls from your S3 site
- c) To speed Lambda cold starts
- d) To secure IAM roles
  Answer: b

24. What can API Gateway generate to ease frontend integration?

- a) S3 website template
- b) JS SDK for your API
- c) IAM user
- d) ACM certificate
  Answer: b

25. The Swagger/OpenAPI spec in the starter repo is intended to:

- a) Auto-train Lex
- b) Import into API Gateway to scaffold routes
- c) Seed DynamoDB
- d) Configure SES templates
  Answer: b

26. Best place to **sign** requests from frontend to API Gateway?

- a) Inside the browser manually
- b) API Gateway generated SDK handles it
- c) Dynamically in SES
- d) In SQS
  Answer: b

27. If frontend can't call API due to blocked preflight:

- a) Disable HTTPS
- b) Enable CORS on methods
- c) Lower Lambda memory
- d) Remove IAM auth
  Answer: b

28. LF0 primarily belongs to which category?

- a) Queue worker
- b) Sync request handler for chat API
- c) Batch processor
- d) Stream consumer
  Answer: b

29. The **frontend host** for the app per assignment is:

- a) EC2
- b) EKS
- c) S3 static website hosting
- d) CloudFront only
  Answer: c

30. Why keep LF0 thin?

- a) Reduce latency & cost; heavy work is async via SQS/LF2
- b) To avoid IAM policies
- c) To bypass logging
- d) To store secrets in code
  Answer: a

D) SQS, SES & Asynchronous Worker (LF2) (10)

31. After Lex collects parameters, where are they enqueued?

- a) SNS
- b) SQS (Q1)
- c) Kinesis
- d) EventBridge bus
  Answer: b

32. LF2, the “suggestions worker,” is triggered:

- a) By user hitting refresh
- b) On a schedule (every minute) to poll Q1
- c) Only by API Gateway
- d) Only by DynamoDB streams
  Answer: b

33. LF2 sends results to the user via:

- a) SMS only
- b) SES email
- c) S3 object events
- d) CloudWatch logs
  Answer: b

34. Which is **not** a property of SQS?

- a) Decoupling producers/consumers
- b) At-least-once delivery
- c) Automatic email sending
- d) Visibility timeout
  Answer: c

35. Which failure policy is encouraged by the assignment?

- a) Drop failed messages silently
- b) Retry forever without limit
- c) Use DLQ after maxReceiveCount exceeded
- d) Retry only once
  Answer: c

36. Why use SQS between Lex and LF2?

- a) To lower EC2 spend
- b) To buffer traffic & smooth spikes
- c) To enforce HTTPS
- d) To compress messages
  Answer: b

37. If SES send fails temporarily, LF2 should:

- a) Delete the message immediately
- b) Keep message so SQS can retry; let it move to DLQ after retries
- c) Panic exit
- d) Replace email with SMS
  Answer: b

38. For traceability of failures, the assignment asks you to:

- a) Delete CloudWatch logs
- b) Log requestId & error reason
- c) Turn off logging
- d) Store logs in S3 only
  Answer: b

39. Benefit of scheduled polling (every minute) for LF2:

- a) Guarantees zero cost
- b) Simplifies autoscaling logic; periodic work cadence
- c) Removes need for queues
- d) Replaces IAM
  Answer: b

40. If Q1 grows faster than LF2 processes:

- a) Add more memory to LF0
- b) Increase LF2 concurrency or shorten schedule interval
- c) Remove DLQ
- d) Decrease SQS retention
  Answer: b

E) DynamoDB (yelp-restaurants) (10)

41. Table name required by assignment:

- a) restaurants-manhattan
- b) yelp-restaurants
- c) dining-concierge
- d) suggestions
  Answer: b

42. Required attribute to attach on each item:

- a) `updatedAt`
- b) `insertedAtTimestamp`
- c) `createdBy`
- d) `ttl`
  Answer: b

43. Why is DynamoDB suitable for scraped restaurant data?

- a) Enforces strict schemas only
- b) Accepts variable attributes per item easily
- c) Requires complex joins
- d) Demands single partition
  Answer: b

44. Which fields are listed as necessary for recommendations?

- a) Business ID, Name, Address, Coordinates, NumReviews, Rating, Zip
- b) Cuisine only

- c) Phone, Website only

- d) Menu file only
  Answer: a

45. How is LF2 expected to use DynamoDB?

- a) To fetch richer details by RestaurantID

- b) To send emails

- c) To replace ElasticSearch

- d) To host frontend
  Answer: a

46. A GSI in this system would be useful to:

- a) Store binary images

- b) Query alternative access patterns (e.g., by cuisine or zip)

- c) Replace SQS

- d) Disable strong consistency
  Answer: b

47. To avoid duplicates while scraping you should:

- a) Use a composite key design or dedupe on Business ID

- b) Disable retries

- c) Delete table daily

- d) Use strong consistency only
  Answer: a

48. If you need time-based housekeeping later, you could:

- a) Use TTL on an epoch attribute

- b) Use S3 lifecycle

- c) Use SES templates
- d) Use ACM certificates
  Answer: a

49. Best pattern to read **many** item details by RestaurantIDs?

- a) `BatchGetItem`
- b) Strongly consistent `GetItem` one by one
- c) `Scan` whole table
- d) `Query` with random key
  Answer: a

50. A hot partition in DynamoDB is usually caused by:

- a) Even key distribution
- b) Highly skewed partition key values
- c) Too many GSIs
- d) Low read capacity only
  Answer: b

F) ElasticSearch / OpenSearch (10)

51. Index requested by the assignment:

- a) `dining`
- b) `restaurants`
- c) `suggestions`

- d) `nyc-food`
  Answer: b

52. Document type (per assignment's wording) under that index:

- a) `Cuisine`
- b) `Restaurant`
- c) `_doc` only
- d) `Place`
  Answer: b

53. What minimal fields must you store there?

- a) RestaurantID & Cuisine
- b) Name & Address
- c) Coordinates & Rating
- d) Zip & Phone
  Answer: a

54. Main purpose of ElasticSearch in this assignment:

- a) Be the system of record
- b) Fast lookup/filter by cuisine, return random IDs
- c) Render the UI
- d) Replace SQS
  Answer: b

55. Why not store full restaurant details only in ES?

- a) ES doesn't support text
- b) Assignment says ES holds a subset; full details come from DynamoDB

- c) ES is too slow

- d) SES requires DynamoDB
  Answer: b

56. If ES returns multiple RestaurantIDs for a cuisine, LF2 should:

- a) Pick a random subset and enrich via DynamoDB

- b) Email all IDs directly

- c) Store them in S3 logs

- d) Call LF0 again
  Answer: a

57. Typical query optimization in ES here:

- a) Use wildcard `*` for everything

- b) Index normalized cuisine keywords

- c) Disable analyzers

- d) Use `scan` only
  Answer: b

58. If cuisine queries are slow, first step:

- a) Increase Lambda memory

- b) Check index mapping/analyzers and add keyword fields

- c) Turn off CORS

- d) Use SES batch send
  Answer: b

59. In a managed OpenSearch domain, scaling reads is often done by:

- a) Adding shards/replicas

- b) Removing nodes

- c) Turning off indexing
- d) Deleting old indices blindly
  Answer: a

60. If ES returns an empty result for a popular cuisine:

- a) Dedupe again
- b) Check whether data was indexed in `restaurants` and mapping is correct
- c) Increase SQS visibility timeout
- d) Repeat `Scan` on DynamoDB
  Answer: b

G) CloudWatch/EventBridge & Scheduling (10)

61. The assignment's automation for LF2 is:

- a) SNS subscription
- b) CloudWatch/EventBridge schedule every minute
- c) API route
- d) Step Functions
  Answer: b

62. A schedule that fires too frequently may cause:

- a) Lower latency only
- b) More Lambda invocations and cost
- c) More DLQ entries automatically

- d) Reduced throughput
  Answer: b

63. If LF2 is idempotent, repeated triggers on empty Q1 will:

- a) Process duplicates
- b) Do nothing (no messages) with minimal harm
- c) Corrupt ES index
- d) Delete the queue
  Answer: b

64. Where should operational logs go?

- a) Nowhere (to reduce cost)
- b) CloudWatch Logs
- c) S3 only
- d) Frontend console
  Answer: b

65. If LF2 errors spike at night, how to inspect?

- a) CloudWatch metrics & logs, view errors and throttles
- b) Delete logs to reduce noise
- c) Turn off DLQ
- d) Scale LF0
  Answer: a

66. Which is **best** for “run every minute”?

- a) EventBridge rule (rate expression)
- b) SQS native scheduler
- c) API Gateway authorizer

- d) SES template
  Answer: a

67. If CloudWatch rule is disabled accidentally, impact:

- a) LF0 fails
- b) LF2 stops polling and emails won't be sent
- c) Lex cannot classify intents
- d) DynamoDB deletes items
  Answer: b

68. What metric reveals queue backlog?

- a) Invocations
- b) ApproximateNumberOfMessagesVisible
- c) 5xx count
- d) Error count
  Answer: b

69. A sudden spike of DLQ messages suggests:

- a) SES or enrichment failures persisting past retries
- b) Healthy system
- c) Missing CORS
- d) Faster ES queries
  Answer: a

70. For proactive alerting on errors:

- a) Create CloudWatch alarms on error metrics and DLQ depth
- b) Turn off logs
- c) Remove IAM policies

- d) Disable retries
  Answer: a

H) End-to-End Architecture Flow (10)

71. The high-level product you're building per assignment:

- a) A video encoder
- b) Dining Concierge chatbot
- c) CI/CD pipeline
- d) Mobile game
  Answer: b

72. Typical user flow (simplified) is:

- a) Frontend → API Gateway → LF0 → Lex → (SQS) → LF2 → ES/DDB → SES
- b) User → SES → LF2 → SQS → Lex
- c) Frontend → DynamoDB → ES → SES
- d) API Gateway → CF → S3
  Answer: a

73. Why decouple Lex from suggestions?

- a) Reduce Lex costs only
- b) Allow async processing and independent scaling
- c) To remove Lambda
- d) To avoid IAM
  Answer: b

74. Where does "collect parameters" happen?

- a) ES
- b) Lex dialog (with Lambda hook)
- c) SES
- d) S3
  Answer: b

75. Where do you **confirm to the user** “we’ll email you suggestions”?

- a) In Lex response after enqueuing to SQS
- b) In SES template
- c) In LF2 logs only
- d) In ES response
  Answer: a

76. Where are raw restaurant documents stored?

- a) DynamoDB (yelp-restaurants)
- b) ES only
- c) S3 only
- d) Lex session
  Answer: a

77. Where are **partial** fields for search stored?

- a) SES
- b) ES index `restaurants` with type `Restaurant`
- c) API Gateway authorizer
- d) S3 website
  Answer: b

78. Which service sends the final email to the user?

- a) SES
- b) SNS
- c) S3
- d) CloudFront
  Answer: a

79. What does LF2 do **before** sending email?

- a) Randomly select IDs from ES for the requested cuisine and enrich via DynamoDB
- b) Query Lex directly
- c) Call API Gateway
- d) Train new intents
  Answer: a

80. What user-visible promise is part of the bot dialog?

- a) “Expect suggestions shortly via email.”
- b) “We’ll text you next week.”
- c) “System under maintenance.”
- d) “We need your password.”
  Answer: a

I) Troubleshooting & Scalability (10)

81. After idle time, high first-request latency in serverless API is most likely:

- a) API Gateway cache misconfig
- b) Lambda cold start

- c) Lex using too many slots
- d) Lambda in a VPC without internet
  Answer: b

82. Sudden surge → immediate failed requests; most likely?

- a) Lambda memory too low
- b) Lambda concurrency limit reached
- c) DynamoDB not optimized
- d) Missing API usage plan
  Answer: b

83. Lex keeps falling back; which **single** fix helps most often?

- a) Add/train more diverse sample utterances for the intent
- b) Disable Lambda hook
- c) Add CORS
- d) Increase API memory
  Answer: a

84. ES queries are slow; first check:

- a) Index mapping/analyzers for cuisine field
- b) API Gateway authorizer
- c) SES identity
- d) S3 bucket region
  Answer: a

85. LF2 timeouts processing messages; first mitigation:

- a) Increase LF2 timeout/memory, optimize downstream calls
- b) Disable retries

- c) Delete DLQ
- d) Remove ES
  Answer: a

86. SQS backlog growing; best action:

- a) Raise LF2 concurrency or frequency of schedule
- b) Lower Lex confidence
- c) Turn off CORS
- d) Use smaller SES batch size
  Answer: a

87. Emails not received; you should check:

- a) SES identity verification and region; logs show error
- b) Lex aliases
- c) ES shards only
- d) DynamoDB hot partition only
  Answer: a

88. “CORS error” in browser console for API calls:

- a) Fix at SES
- b) Enable CORS on API methods
- c) Add DLQ
- d) Increase Lambda memory
  Answer: b

89. Duplicate restaurants appear in emails; fix:

- a) Enforce dedupe by Business ID (during scrape or before email)
- b) Ask Lex to dedupe

- c) Use SES dedupe
- d) Set SQS delay
  Answer: a

90. LF0 returning HTTP 500 intermittently; likely cause:

- a) Lex call errors or timeouts; check retries/logging
- b) SES identity blocked
- c) S3 website disabled
- d) ES cluster red
  Answer: a

J) Extra Credit DLQ, Data, & Best Practices (10)

91. The assignment's DLQ direction is to:

- a) Attach DLQ to Q1 with maxReceiveCount (e.g., 3–5)
- b) Use DLQ on SES
- c) Replace SQS with DLQ
- d) Use DLQ on API Gateway
  Answer: a

92. On SES failure, worker behavior should be:

- a) Delete message from Q1
- b) Keep message; allow retry; move to DLQ after limit
- c) Disable logging

- d) Re-index ES
  Answer: b

93. Evidence for DLQ correctness should include:

- a) A tweet
- b) Screenshot of message in DLQ + CloudWatch log entry
- c) ES query time screenshot
- d) S3 listing
  Answer: b

94. Why store `insertedAtTimestamp`?

- a) Sorting/TTL/ops observability on ingest time
- b) SES requirement
- c) ES mapping constraint
- d) API Gateway limit
  Answer: a

95. If you need 1,000+ restaurants with **≥5 cuisines**, the assignment suggests:

- a) Yelp API; ~200 per cuisine; avoid duplicates
- b) OpenStreetMap only
- c) Manual CSV upload
- d) Hard-coded list
  Answer: a

96. Best place to keep the **frontend**?

- a) S3 static hosting (optionally behind CloudFront)
- b) SES
- c) ES domain

- d) DynamoDB
  Answer: a

97. What does the API Gateway **JS SDK** mainly give you?

- a) DynamoDB CRUD UI
- b) Signed API calls & easy client methods
- c) SES email templates
- d) Lex bot trainer
  Answer: b

98. Which piece is **explicitly** not part of neighborhood filtering per assignment?

- a) Filter by cuisine only; neighborhood not required
- b) Filter by zip strictly
- c) Filter by rating strictly
- d) Filter by price strictly
  Answer: a

99. What is the **core** architectural style encouraged?

- a) Monolith on EC2
- b) Serverless, microservice-driven app
- c) On-prem hosted cluster
- d) Stateful Lambdas
  Answer: b

100. The “glue” that turns user chat into emailed suggestions is mainly:

- a) Direct LF0 emails
- b) Lex + SQS (Q1) + LF2 + ES/DDB + SES
- c) ES only

- d) Frontend only
  Answer: b

If your **Docker container for MongoDB does not specify a persistent volume**, then:

All your data will be lost when the container stops, restarts, or is removed.

Explanation:

- By default, container data is stored **ephemerally** inside the container's writable layer.
- When the container is deleted or rebuilt (e.g., via `docker rm`, `docker-compose down`, or image update), that layer disappears — taking your MongoDB data with it.
- Without a persistent volume, MongoDB's `/data/db` directory has **no external mount**, so it cannot retain database files across restarts.

Correct preventive step:

✅ Use a **persistent or named volume**:

```
docker run -d \
  --name mongo \
  -v mongodb_data:/data/db \
  mongo
```

This ensures that your database files survive container restarts and redeployments.