Question 1 of 15

Web servers and browsers impose payload size limits that prevent large files from being uploaded in a single HTTP request.

1 True

2 False

Correct!

Most web servers (Apache, NGINX) default to ~2GB request limits, while services like API Gateway often limit requests to 10MB or less. Browsers also impose their own limits. These hard constraints make chunking necessary for large file uploads, not just beneficial.

Question 2 of 15

Which technology distributes content geographically to reduce latency?

1 Content Delivery Network (CDN)

2 Load Balancer

3 Message Queue

4 Database Replica

Correct!

CDNs cache content at edge locations worldwide, serving users from the nearest geographic point. This reduces network distance and latency compared to serving all content from a single location.

Question 3 of 15

Which is NOT a benefit of horizontally scalable storage systems?

1 Fault tolerance

2 Guaranteed consistency

3 Virtually unlimited capacity

4 Automatic redundancy

Correct!

Horizontally scalable storage systems typically sacrifice strong consistency for availability and partition tolerance (CAP theorem). They provide capacity, redundancy, and fault tolerance, but not guaranteed immediate consistency across all nodes.

Question 4 of 15

Presigned URLs provide temporary access to storage resources without permanent permissions.

1 True

2 False

Correct!

Presigned URLs include time-limited authentication tokens that grant temporary access to specific resources. This security pattern allows controlled access without modifying permanent ACLs or sharing long-term credentials.

Question 5 of 15

For file storage systems, which CAP theorem property is typically prioritized?

1 All equally

2 Availability

3 Partition Tolerance

4 Consistency

Correct!

File storage systems typically prioritize availability over consistency because users expect to access their files even during network partitions. Temporary inconsistency (like a file taking seconds to appear on another device) is acceptable.

Question 6 of 15

Compression always improves upload performance regardless of file type.

1 True

2 False

Correct!

Compression only improves performance when the time saved from transferring fewer bytes exceeds the time spent compressing. Already-compressed files (images, videos) see minimal size reduction, making compression counterproductive.

Question 7 of 15

A system needs to upload 50GB files reliably. Which approach works best?

1 Chunked multipart upload

2 Direct database insertion

3 Email attachment

4 Single HTTP POST request

Correct!

Chunked multipart upload breaks large files into manageable pieces, enabling progress tracking, resumability on failure, and avoiding timeout issues that plague single large requests.

Question 8 of 15

Which technique enables content-based file deduplication and identification?

1 Directory structure

2 Cryptographic fingerprinting

3 Metadata tagging

4 File naming

Correct!

Cryptographic fingerprinting (hashing) creates unique identifiers based on file content, enabling systems to detect duplicate files regardless of their names or locations. This supports both deduplication and resumable uploads.

Question 9 of 15

Parallel chunk uploads can maximize available bandwidth utilization.

1 True

2 False

Correct!

Parallel uploads utilize multiple connections simultaneously, allowing systems to saturate available bandwidth more effectively than sequential transfers. This is especially beneficial for high-latency or high-bandwidth connections.

Question 10 of 15

When a file upload finishes, what's the most reliable method to update the file's metadata with its completion status?

1 Client sends update request

2 Manual verification process

3 Storage service triggers event notification

4 Periodic polling for status changes

Correct!

Storage services like S3 provide event notifications (via SNS/SQS) when uploads complete. These events are reliable because they come directly from S3 after it confirms the upload is complete, handling cases where clients crash or disconnect after upload. While client notifications are simpler to implement, they're less reliable since clients can fail to send the update, leaving metadata inconsistent. Storage event notifications provide the most robust solution for production systems despite added complexity.

Question 11 of 15

Which sync approach optimizes both performance and resource usage?

1 Manual sync only

2 WebSocket connections for all files

3 Continuous polling for all files

4 Hybrid: real-time for active, polling for stale

Correct!

Hybrid approaches balance performance and resource efficiency by using expensive real-time connections only for frequently accessed files while falling back to cheaper polling for inactive files.

Question 12 of 15

Signed URLs with expiration timestamps become unusable after their time limit, even if shared with authorized users.

1 True

2 False

Correct!

Time-limited URLs include expiration timestamps in their cryptographic signatures. After expiration, the signature becomes invalid, preventing access regardless of who has the URL.

Question 13 of 15

During network failure, resumable uploads require which information to continue?

1 Original file path only

2 Chunk upload completion status

3 Network diagnostics

4 User credentials only

Correct!

Resumable uploads need to track which chunks have been successfully transferred to avoid re-uploading completed portions. This state information enables efficient recovery from interruptions.

Question 14 of 15

Which factor most impacts download speed for globally distributed users?

1 Database query speed

2 Server CPU power

3 Geographic distance to content

4 User device type

Correct!

Network latency increases significantly with geographic distance due to the speed of light limitations. CDNs address this by caching content closer to users, reducing the physical distance data must travel.

Question 15 of 15

Last-write-wins strategy ensures strong data consistency in distributed file systems.

1 True

2 False

Incorrect.

Last-write-wins provides eventual consistency, not strong consistency. It resolves conflicts by accepting the most recent update, but doesn't prevent concurrent modifications or guarantee immediate consistency across all nodes.