Interview questions - Object Storage in Modern Systems

💬 Conceptual Questions

1. What is object storage and how is it different from file or block storage?

- **Object Storage** stores data as discrete units (objects), each with a unique identifier and rich metadata. These objects are stored in a flat address space (buckets).
- **File Storage** uses a hierarchical structure (folders/directories). It's good for shared file systems.
- **Block Storage** breaks data into fixed-sized blocks without metadata. Often used in databases or virtual machines.
- **Key difference:** Object storage is **ideal for scalability, metadata support, and unstructured data**, while file/block storage is better for low-latency or transactional workloads.

2. When would you choose object storage over a traditional file system?

- When dealing with large amounts of **unstructured data** (e.g., media, backups, logs).
- When **scalability** is critical — object storage is built to scale across regions.
- When you need to **serve static assets** like images, videos, or documents over the internet.
- When **cost efficiency** over long-term storage is more important than real-time performance.

3. Explain the structure of an object in object storage. What role does metadata play?

- An object consists of:
    - The **data** (e.g., image file).
    - A **unique ID** or key.

    - **Custom metadata** (user-defined or system-generated) — e.g., file type, permissions, timestamps, tags.
- **Metadata** enables powerful organization, filtering, lifecycle policies, and search mechanisms.

🧠 System Design Scenarios

4. Design a media hosting platform (e.g., YouTube). How would you use object storage for video uploads and streaming?

- Use object storage (like S3) to store uploaded videos.
- Organize by user ID or content type using bucket prefixes or tags.
- Use **presigned URLs** for secure upload/download.
- Integrate with **CDNs** (CloudFront, Akamai) for faster video delivery.
- Store **different video resolutions as separate objects** to support adaptive streaming.

5. How would you architect a cost-efficient backup system for petabytes of logs using Amazon S3?

- Use **S3 lifecycle rules** to transition logs from:
    - S3 Standard → S3 Infrequent Access → Glacier → Glacier Deep Archive.
- Enable **versioning** and **object lock** for immutability.
- Use **batch processing** (e.g., AWS Batch or Lambda) to compress and archive logs.
- Monitor access patterns and optimize class usage with **S3 Storage Class Analysis**.

6. In a microservices system, how would services securely share large files using object storage?

- Upload files to object storage and generate **presigned URLs** for limited-time access.
- Apply **IAM policies** for fine-grained access control.
- Encrypt objects using **SSE-S3** or **SSE-KMS**.

- Track object usage or audit via **CloudTrail** or S3 access logs.

⚙️ Practical & Trade-off Questions

7. What are the performance trade-offs of using object storage for real-time access?

- **Higher latency** than block storage (especially for small, random reads/writes).
- **Eventual consistency** for overwrite or delete operations in some regions.
- Not suitable for **real-time transactional systems** — better for cold or warm data.

8. How do storage class tiers (e.g., S3 Standard vs. Glacier) influence design decisions?

- **S3 Standard**: High availability and low latency — used for frequently accessed content.
- **S3 IA / One Zone IA**: Cheaper but with retrieval fees — used for infrequently accessed backups.
- **S3 Glacier / Deep Archive**: Very cheap but high retrieval latency — ideal for compliance and archival.
- **Trade-off**: Cost vs. access time — you need lifecycle management to optimize this.

9. How would you implement access control for user-specific data stored in object storage?

- **Bucket policies** or **IAM roles** scoped to individual users or applications.
- Use **object-level permissions** or **presigned URLs** for user uploads/downloads.
- Enable **logging and monitoring** to detect unauthorized access attempts.
- Encrypt objects and manage keys using **KMS** or customer-managed encryption keys (CMKs).