[**https://github.com/minio/minio/issues/21603**](https://github.com/minio/minio/issues/21603)

**Conditional PUTs and Read Quorum**

**Question 3**

Conditional PUTs ignore their conditions when read quorum cannot be reached. Why does this happen in MinIO?

Background

MinIO is a distributed object storage system. When running in distributed mode:

Each object is stored across multiple nodes using erasure coding.

Operations like PUT (write) and GET (read) rely on quorum rules to maintain data consistency.

Key concepts:

Read Quorum

Minimum number of nodes that must respond successfully to consider a read valid.

Example: If object is split across 6 disks (4 data + 2 parity), the read quorum might be 3.

Write Quorum

Minimum number of nodes that must acknowledge a write for it to be successful.

Helps ensure data durability.

Conditional PUT

A PUT request with a condition (like If-Match or If-None-Match) checks an ETag or version before writing.

Example: Only overwrite if object version equals a specific ETag.

MinIO cannot verify the object state

Conditional PUT depends on the current object’s ETag or version.

If the read quorum fails (some nodes are down/unavailable), MinIO cannot reliably check the condition.

**Fail-safe Behavior**

To avoid blocking writes indefinitely, MinIO allows the PUT to proceed, ignoring the condition.

This ensures availability over strict consistency in distributed mode.

Trade-off: Availability vs Consistency

MinIO follows an AP-like behavior (Availability over strict Consistency) in distributed setups.

Conditional checks are best-effort, but if quorum is missing, availability wins.

**Conclusion**

When you do a conditional PUT in MinIO (like “only overwrite this file if it matches a version”), the server first needs to check the current state of the object by reading it from enough nodes — this is called the read quorum.

If MinIO cannot reach enough nodes to reliably read the object, it cannot verify the condition. To avoid blocking the write completely, MinIO ignores the condition and proceeds with the upload.

If MinIO can’t read enough copies of the object to check your condition, it prioritizes allowing the write rather than failing it.