**🔹 1. POST /files → Presigned URL**

**Why POST?**

* POST is used to **create a new resource** — in this case, a new file upload session or request.
* It is **non-idempotent**, meaning multiple POSTs could result in multiple presigned URLs or upload sessions.
* You're **submitting metadata** about a file, and the server responds with a **newly created upload link** (a presigned URL).

✅ Appropriate use of POST to **initiate creation** of something.

**🔹 2. PUT {Presigned URL} → Upload file chunk**

**Why PUT?**

* PUT is used to **upload or replace a resource at a specific URL** — in this case, the presigned URL points to a storage location (like S3).
* It is **idempotent** — uploading the same file chunk multiple times will result in the same final state.
* It conveys: “upload this file data **to this exact location**.”

✅ Perfect use of PUT: You are placing a resource (file chunk) **at a known location**.

**🔹 3. PATCH /files → Update metadata (e.g., chunk status)**

**Why PATCH?**

* PATCH is used to **partially update** a resource.
* You’re only updating certain fields (e.g., "chunk 1 uploaded"), not the whole file metadata.
* PATCH is idempotent (usually) and efficient for **incremental status updates**.

✅ Proper use of PATCH to **modify just parts** of a resource.

**🔹 4. GET /files/{fileId} → Get file & metadata**

**Why GET?**

* GET is used to **retrieve data** without modifying anything.
* It is **safe and idempotent** — calling it multiple times doesn’t change anything.

✅ Standard use of GET to **fetch** a file and its metadata.

**🔹 5. GET /changes?since={timestamp} → Get updated file metadata**

**Why GET?**

* Again, you're **retrieving** data — in this case, changes that happened since a certain time.
* The server doesn’t change any state as a result of this call.