2. Source control strategy

└ 2.2 Repository Configuration and Maintenance

└ 2.2.1 Manage large Files (GIT LFS, git-fat)

---

1. What is Git LFS and why is it needed in source control?
2. How does Git LFS work under the hood?
3. What types of files should be managed using Git LFS?
4. How do you install and initialize Git LFS in a repository?
5. How do you track files with Git LFS and what is the basic workflow?
6. How does storage and bandwidth quota work with Git LFS?
7. What are key differences between Git LFS and git-fat?
8. How do you migrate existing large files in a repository to Git LFS?
9. What are best practices for managing large files in Azure DevOps or GitHub using LFS?
10. How do you handle permissions and security for large files stored in Git LFS?

---

1. **What is Git LFS and why is it needed in source control?**

Git LFS (Large File Storage) is an extension that replaces large files (like binaries, media, datasets) in Git with lightweight pointers, storing the actual files in external storage. It's needed because Git is inefficient with large or frequently changing binary files.

---

2. **How does Git LFS work under the hood?**

When you commit a tracked large file, *Git LFS* stores a pointer in the repo and uploads the actual content to a remote LFS storage. During checkout, LFS downloads the actual files on demand.

---

3. **What types of files should be managed using Git LFS?**

Large binary files, media assets (e.g., images, videos), archives, datasets, or any files that are large or change frequently, and are not efficiently diffed or compressed by Git.

---

4. **How do you install and initialize Git LFS in a repository?**

Install Git LFS with:

```
git lfs install
```

Then track file types:

```
git lfs track "*.psd"
```

Commit the `.gitattributes` file and use Git as normal.

---

5. **How do you track files with Git LFS and what is the basic workflow?**

Track file patterns with *git lfs track*, add files, commit, and push as usual. LFS replaces tracked files with pointers in the repo and uploads file contents to LFS storage during push.

---

6. **How does storage and bandwidth quota work with Git LFS?**

Git LFS hosting providers (e.g., *GitHub*, *Azure DevOps*) enforce storage and bandwidth limits. Exceeding quota can block uploads or downloads until more space or bandwidth is purchased or usage is reduced.

---

7. **What are key differences between Git LFS and git-fat?**

Git LFS is widely supported, natively integrated with *GitHub/Azure DevOps*, and stores large files externally. *git-fat* is older, works with custom backends, and is less integrated into cloud providers.

8. **How do you migrate existing large files in a repository to Git LFS?**

Use `git lfs migrate import --include="*.filetype"` to convert files to LFS tracking, rewrite history as needed, and force-push updated branches. Always back up before migrating.

---

9. **What are best practices for managing large files in Azure DevOps or GitHub using LFS?**
    - Track only necessary files,
    - use `.gitattributes` to specify file patterns,
    - monitor LFS quota,
    - educate contributors,
    - and avoid committing large files directly to Git.

---

10. **How do you handle permissions and security for large files stored in Git LFS?**

Permissions follow the repository's access control (*GitHub*/*Azure DevOps*).
To protect sensitive files,

- restrict repo access,
- use branch protections,
- and regularly audit who can push/pull LFS content.