🧠 Git LFS — One-Page Mental Model + Setup

◇ What Git LFS actually does

- Git **does NOT store big files**
- Git stores a **tiny pointer**
- The real file is stored in **LFS storage**
- This avoids push size limits and repo bloat

◇ Golden rules (memorize these)

1. **Track file TYPES, not files**
2. **Track BEFORE committing**
3. `.gitattributes` is like `.gitignore`
4. Commit `.gitattributes` **only when it changes**
5. After setup → **normal Git workflow**

◇ Files you usually track with LFS

Typical examples:

- Archives: `*.zip`, `*.tar`, `*.gz`, `*.7z`
- Models: `*.bin`, `*.pt`, `*.h5`, `*.onnx`
- Media: `*.mp4`, `*.mov`, `*.wav`
- Data: `*.csv`, `*.parquet`

🚀 Clean Git LFS Setup (DO THIS ONCE)

1️⃣ Install & initialize LFS

```
git lfs install
```

2️⃣ Tell Git which files go to LFS

(Use patterns, not filenames)

```
git lfs track "*.zip"
git lfs track "*.bin"
git lfs track "*.mp4"
```

➡️ This creates/updates `.gitattributes`

3️⃣ Commit LFS rules

```
git add .gitattributes
git commit -m "Configure Git LFS"
```

◇ From now on (normal daily workflow)

```
git add largefile.zip
git commit -m "Add dataset"
git push
```

👉 **No extra LFS commands needed**

🔍 Verify anytime

```
git lfs ls-files
```

If listed → ✅ LFS is working

🔄 If you already committed a big file (fix)

```
git rm --cached bigfile.zip
git add bigfile.zip
git commit -m "Move file to Git LFS"
```

🧹 Full reset (files stay safe)

```
rm -rf .git
git init
git lfs install
git lfs track "*.zip" "*.bin"
git add .
git commit -m "Initial commit with Git LFS"
```

❌ Things that break LFS

- Tracking after committing
- Tracking individual filenames
- Forgetting to commit `.gitattributes`
- Deleting repo before pushing

🧩 Mental shortcut

“If it’s big → pattern → `.gitattributes` → forget about it”