Absolutely. Here's a **comprehensive Git command cheat sheet** tailored to real-world scenarios like yours — including recovering from lost environments, pushing updates, avoiding merge conflicts, and more.

**🧾 Git Command Help Sheet – Practical Scenarios**

**🔰 INIT & SETUP**

| **Task** | **Command** |
| --- | --- |
| Initialize a new Git repo | git init |
| Add a remote origin (GitHub repo) | git remote add origin <URL> |
| Check current remotes | git remote -v |
| Clone a repo from GitHub | git clone <URL> |

**💾 STAGING & COMMITTING**

| **Task** | **Command** |
| --- | --- |
| Stage all changes | git add . |
| Commit changes with message | git commit -m "Your message" |
| See what changed (unstaged/staged) | git status |
| See the diff of unstaged changes | git diff |
| View commit history | git log --oneline --graph --all |

**📤 PUSHING & PULLING**

| **Task** | **Command** |
| --- | --- |
| Push code to GitHub (first time) | git push -u origin main |
| Force push (overwrite GitHub version) | git push --force or git push -f |
| Pull with unrelated histories (e.g. after re-init) | git pull origin main --allow-unrelated-histories |
| Pull latest changes from GitHub | git pull origin main |
| Set the default branch to main | git branch -M main |

**🆘 CONFLICT RECOVERY & SAFE BACKUP**

| **Task** | **Command** |
| --- | --- |
| Stash (temporarily save) your local changes | git stash |
| Apply stashed changes | git stash pop |
| View stash list | git stash list |
| Back up local files (manual) | cp -r ./project ./project\_backup |
| Recover from GitHub by cloning again | git clone <repo-url> |

**⚙️ COMMON ERRORS & FIXES**

| **Issue** | **Fix** |
| --- | --- |
| You have to pull first | git pull --rebase OR git pull --allow-unrelated-histories |
| Remote already exists | git remote set-url origin <new-url> |
| "fatal: refusing to merge unrelated histories" | git pull origin main --allow-unrelated-histories |
| "Repository not found" | Check URL and access permissions |
| GitHub push denied (SSH vs HTTPS conflict) | Use the correct protocol or re-add remote |

**🧪 TESTING & DRY RUNS**

| **Task** | **Command** |
| --- | --- |
| Preview what will be committed | git diff --staged |
| See last commit | git show |
| See what files changed in history | git log --stat |

**🧹 RESETTING & CLEANUP**

| **Task** | **Command** |
| --- | --- |
| Discard all local changes (DANGER) | git reset --hard |
| Remove staged files | git reset |
| Remove untracked files | git clean -f |
| Reset to specific commit | git reset --hard <commit-id> |

**🧰 ADVANCED & SAFETY**

| **Task** | **Command** |
| --- | --- |
| Create a new branch | git checkout -b new-branch |
| Switch branches | git checkout main |
| See current branch | git branch |
| Delete a branch (local) | git branch -d branch-name |
| Tag a release | git tag v1.0 |
| Push tag to remote | git push origin v1.0 |

**🛡️ GITHUB-SPECIFIC COMMANDS**

| **Task** | **Command** |
| --- | --- |
| Push to GitHub with SSH | git remote set-url origin git@github.com:user/repo.git |
| Push to GitHub with HTTPS | git remote set-url origin https://github.com/user/repo.git |
| Authenticate with personal access token | Use https://<username>:<token>@github.com/user/repo.git |
| Change visibility on GitHub | Go to **Settings > Danger Zone > Change visibility** |

**📎 Best Practices**

* Always git status before and after staging.
* Use descriptive commit messages: git commit -m "Fix API auth error"
* Pull before you push if you're unsure: git pull origin main
* Never --force unless you're sure what you're overwriting.

Would you like this in a downloadable **PDF** or **Markdown file**?