# **Push, Pull, and Prosper: The Git Playbook**

Welcome to my “Git Playbook”! This comprehensive guide is your gateway to mastering Git, the essential tool for modern software development. If you're a beginner and looking to learn about version controlling and sharpen your skills, but don’t know where to start, this book will guide you through the impossible.

My lessons will be brief and relevant, **NO JARGON!** Which will provide clear, practical insights to help you navigate and excel in version control.

**Basic Commands**

* **git init**: Initializes a new Git repository.
* **git clone <repository>**: Clones an existing repository from a URL.
* **git status**: Displays the state of the working directory and staging area.
* **git add <file>**: Adds a file to the staging area.
* **git commit -m "message"**: Records changes to the repository with a message.
* **git push**: Uploads local repository content to a remote repository.
* **git pull**: Fetches and integrates changes with another repository or a local branch.
* **git fetch**: Downloads objects and refs from another repository.
* **git merge <branch>**: Merges changes from one specific branch into the current branch.

**Branching and Tagging**

* **git branch**: Lists all local branches.
* **git branch <branch-name>**: Creates a new branch.
* **git checkout <branch-name>**: Switches to a different branch.
* **git checkout -b <branch>**: Creates and switches to a new branch.
* **git tag <tag-name>**: Creates a tag at the current commit with a specified name.
* **git switch**: Switch branches.

**Remote Repositories**

* **git remote**: Lists remote connections.
* **git remote add <name> <url>**: Adds a new remote repository.
* **git remote -v**: Shows URLs of the remote repositories.

**Inspecting History and State**

* **git log**: Shows commit history logs.
* **git diff**: Shows the difference between the working directory and the staging area, or between the staging area and the last commit.
* **git show <commit>**: Shows various types of objects.
* **git blame <file>**: Shows what revision and author last modified each line of a file.

**Undoing Changes**

* **git reset <file>**: Removes file(s) from the staging area.
* **git reset --hard <commit>**: Resets the index and working directory to a specified commit.
* **git restore:** Restore working tree files.
* **git rm [file]:** Removes a file from the staging area and working directory.
* **git revert <commit>**: Creates a new commit that undoes changes from a previous commit.
* **git clean -f**: Removes untracked files from the working directory.
* **git mv <old-file-path> <new-file-path>:** Move or rename a file, directory, or a symlink.

**Stashing**

* **git stash**: Stashes the changes in a dirty working directory away/ Temporarily saves uncommitted changes.
* **git stash pop**: Applies the changes stashed away and removes the stash.
* **git stash list**: Lists all stashes.

**Configuration**

* **git config --global user.name "name"**: Sets the name of the user for all repositories.
* **git config --global user.email "email"**: Sets the email of the user for all repositories.
* **git config --list**: Lists all configuration settings.

**Advanced Commands**

* **git rebase <branch>**: Reapplies commits on top of another base tip.
* **git cherry-pick <commit>**: Applies the changes introduced by some existing commits.
* **git bisect**: Uses binary search to find the commit that introduced a bug.
* **git submodule**: Manages submodules (repositories nested inside a parent repository).
* **git grep**: Print lines matching a pattern.