Git & GitHub Commands Cheat Sheet for DevOps Engineers

---

1. Git Configuration

`git config --global user.name "Your Name"`

Set username.

`git config --global user.email "your@email.com"`

Set email.

`git config --global -l`

Show all global Git configs.

`git config --global core.editor "vim"`

Set default editor.

---

2. Repository Initialization & Cloning

`git init`

Initialize a new Git repository.

`git clone <repo-url>`

Clone a remote repository.

`git clone -b dev <repo-url>`

Clone only the dev branch.

---

3. Basic File Operations

`git status`

Show changes in working directory.

`git add file.txt`

Stage a file.

`git add .`

Stage everything.

`git commit -m "message"`

Commit with a message.

`git commit -am "message"`

Add + commit modified files.

---

4. Branch Operations

`git branch`

List branches.

`git branch dev`

Create new branch.

`git checkout dev`

Switch to dev branch.

`git checkout -b feature/login`

Create + switch to new branch.

`git branch -d dev`

Delete branch locally.

`git push origin --delete dev`

Delete branch remotely.

---

5. Pull, Fetch & Merge

`git pull`

Fetch + merge from remote.

`git fetch`

Download changes without merging.

`git merge dev`

Merge dev into current branch.

---

6. Merge Conflict Commands

`git merge dev`

May cause conflict.

`git status`

Shows conflicting files.

`git add <file>`

Stage resolved file.

`git commit`

Finalize merge once resolved.

---

7. Rebase Commands (Important for DevOps)

`git rebase main`

Rebase your branch on top of main branch.

`git rebase -i HEAD~5`

Interactive rebase (edit, squash commits).

`git rebase --continue`

Continue after conflict resolution.

`git rebase --abort`

Abort the rebase process.

---

8. Stash Commands

`git stash`

Save uncommitted changes temporarily.

`git stash pop`

Apply stash and remove from stash list.

`git stash apply`

Apply stash without removing.

`git stash list`

Show stashes.

`git stash drop`

Delete a stash.

---

9. Log & History Commands

`git log`

View commit history.

`git log --oneline --graph --decorate`

Visual commit tree.

`git show <commit>`

See details of a specific commit.

`git diff`

Compare changes not staged.

`git diff --staged`

Compare staged changes.

---

10. Remote Repository Commands

`git remote -v`

Show remotes.

`git remote add origin <url>`

Add remote repo.

`git push origin main`

Push code to main branch.

`git push -u origin dev`

Push dev branch & set upstream.

---

11. Remove & Restore Files

`git rm file.txt`

Remove file from repo & disk.

`git rm --cached file.txt`

Remove file from repo but keep on disk.

`git restore file.txt`

Restore a deleted or changed file.

`git restore --staged file.txt`

Unstage a file.

---

12. Tags (Release Management)

`git tag v1.0`

Create a tag.

`git tag`

List tags.

`git tag -a v1.0 -m "release 1.0"`

Annotated tag.

`git push origin v1.0`

Push tag to GitHub.

---

13. GitHub Pull Request (PR)

Steps (Important for DevOps):

1. Push branch → `git push origin feature/login`
2. Go to GitHub → Create Pull Request
3. Add reviewers
4. Resolve comments
5. Merge PR (Squash/Merge/Rebase)
6. Delete branch

---

14. Squash Multiple Commits

`git rebase -i HEAD~4`

Use `s` to squash commits.

`git commit --amend`

Modify the last commit message.

---

15. Cherry Pick (VERY important for DevOps)

`git cherry-pick <commit-id>`

Apply a specific commit from one branch to another.

---

16. Undo & Rollback

`git reset --soft HEAD~1`

Undo last commit but keep changes staged.

`git reset --hard HEAD~1`

Undo last commit & discard changes.

`git revert <commit>`

Create a new commit that reverses a change.

---

17. Clean Working Directory

`git clean -n`

Preview what will be removed.

`git clean -f`

Force delete untracked files.

`git clean -fd`

Delete untracked files + folders.

---

18. GitHub SSH Setup

`ssh-keygen -t rsa`

Generate SSH key.

`cat ~/.ssh/id_rsa.pub`

Copy key.

Add key to GitHub → Settings → SSH Keys.

---

19. Forking Workflow

`git clone <forked-repo>`

Clone your fork.

`git remote add upstream <original-repo>`

Add main/original repo.

`git fetch upstream`

Get latest changes.

`git merge upstream/main`

Sync with original repo.

---

20. GitHub Actions (DevOps Essential)

`.github/workflows/ci.yml`

Location of GitHub Actions workflow file.

Trigger pipeline manually:

```
workflow_dispatch:
```

Trigger on push:

```
on:
  push:
    branches: [ main ]
```

---

21. Git Submodules

`git submodule add <url> path/`

Add a submodule.

`git submodule update --init --recursive`

Initialize submodules.

`git submodule sync`

Sync submodule URLs.

---

22. Git Bisect (Bug Hunting)

`git bisect start`

Start binary search.

`git bisect good <commit>`

Mark commit as good.

`git bisect bad <commit>`

Mark commit as bad.

`git bisect reset`

End bisect.

---

23. Git Archive (Create Zip of Repo)

`git archive --format zip --output=repo.zip main`

Create zip of repository.

---

24. GitHub CLI (gh)

If installed:

`gh repo clone owner/repo`

Clone repo using GitHub CLI.

`gh pr create`

Create PR from terminal.

`gh pr list`

List pull requests.

25. Branch Tracking & Upstream Commands

`git branch -vv`

Shows branches + their upstream (tracking) branches.

`git push -u origin dev`

Sets the upstream for dev branch.

`git branch -u origin/main`

Link your current branch to remote main.

`git rev-parse --abbrev-ref --symbolic-full-name @{u}`

Show the upstream branch of current branch.

---

26. Advanced Fetch Commands

`git fetch --all`

Fetch changes from all remotes.

`git fetch origin main`

Fetch only main branch updates.

`git fetch -p`

Prune deleted remote branches.

---

27. Advanced Merge Techniques

`git merge --abort`

Abort merge when conflicts occur.

`git merge --no-ff dev`

Force a merge commit (important for clean history).

`git merge --squash dev`

Squash branch commits into one without auto merge.

---

28. Working With Detached HEAD

`git checkout <commit-id>`

Checkout a commit → enters detached HEAD state.

`git switch -c new-branch`

Save the detached state into a new branch.

---

29. Compare Branches & Commits

`git diff main..dev`

Compare two branches.

`git diff <commit1> <commit2>`

Compare two commits.

`git log main..dev`

Show commits missing in main compared to dev.

---

30. Git Patch Files

`git format-patch -1`

Create patch for last commit.

`git apply file.patch`

Apply a patch file.

`git am file.patch`

Apply a series of patches.

---

31. Git Blame (Debugging Code Changes)

`git blame file.py`

Show who changed which line.

`git blame -L 10,30 app.js`

Blame lines 10–30.

---

32. Git Clean & Workspace Reset

`git restore .`

Restore all modified files.

`git reset --hard`

Reset to last commit (removes changes).

`git reset --hard origin/main`

Reset local branch to remote main.

---

33. Git Reflog (Most Powerful Recovery Tool)

`git reflog`

Shows entire command history (can recover lost commits, branches).

`git checkout <reflog-id>`

Restore deleted work using reflog.

---

34. Git Garbage Collection

`git gc`

Optimize repository.

`git gc --prune=now`

Remove all unreachable objects immediately.

---

35. Git Notes

`git notes add -m "comment"`

Attach notes to a commit (used in CI/CD tagging).

`git notes show`

View commit notes.

---

36. Git Hooks (Automation Before Commit/Push)

Located in `.git/hooks/`

`pre-commit`

Runs before committing (used for linting, formatting).

`pre-push`

Runs before pushing code (used for tests).

Disable hook temporarily:

`git commit --no-verify`

Skip hooks.

---

37. Git Ignore Management

`.gitignore`

Files ignored by Git.

`git rm -r --cached .`

Remove all files from Git tracking (useful after updating .gitignore).

---

38. Git LFS (Large File Storage)

`git lfs install`

Enable Git LFS.

`git lfs track "*.zip"`

Track large file types.

`git lfs push origin main`

Push large files using LFS.

---

39. GitHub Security Features

`.github/dependabot.yml`

Automated dependency updates.

`gh secret set AWS_ACCESS_KEY`

Add GitHub Actions secret.

Remove secret:

```
gh secret delete AWS_ACCESS_KEY
```

---

40. GitHub Pages

Enable GitHub Pages:

Settings → Pages → Select branch

Create branch for pages:

```
git checkout -b gh-pages
```

---

41. GitHub Releases

Create release:

GitHub → Releases → New Release

Upload artifacts or binaries.

---

42. Fork Syncing (Advanced)

`git fetch upstream`

Fetch from main/original repo.

`git rebase upstream/main`

Rebase your fork.

---

43. Multi-Remote Setup

`git remote add staging <repo-url>`

Add second remote.

`git push staging dev`

Push code to staging remote.

`git remote rename origin production`

Rename remote.

---

44. GPG Commit Signing

`gpg --list-secret-keys --keyid-format LONG`

Find Key ID.

`git config --global commit.gpgsign true`

Enable commit signing.

---

45. Git Credential Store

`git config --global credential.helper store`

Save credentials locally.

`git credential reject`

Remove stored credentials.

---

46. Git Archive (CI/CD Deployments)

`git archive HEAD | tar -x -C /deploy/dir`

Extract repo contents to deployment folder.

---

47. Git Worktree (Multiple Working Copies)

`git worktree add ../temp feature2`

Have two branches checked out simultaneously.

`git worktree list`

List active linked worktrees.

---

48. Git Bisect (Find Bug-Causing Commit)

`git bisect start`

Start bisect.

`git bisect good <commit>`

Mark commit as good.

`git bisect bad <commit>`

Mark commit as bad.

---

49. GitHub CLI (More Advanced)

`gh repo view`

View GitHub repo details.

`gh issue list`

List issues.

`gh workflow run build.yml`

Trigger GitHub Actions workflow manually.

---

50. GitHub API via Curl

`curl -H "Authorization: token <TOKEN>" https://api.github.com/user/repos`

List GitHub repos programmatically.
