Git Cheat Sheet

Hochschule Darmstadt, Fachbereich Elektro- und Informationstechnik

Konfiguration

```
git config --global user.name "Your Name"
git config --global user.email "your_email@h-da.de"
git config --global core.editor nano  # oder vi
```

Repository anlegen

```
git init               # Repo im aktuellen Verzeichnis anlegen
git checkout -b main   # Neuen Branch 'main' erstellen und wechseln
```

Dateien zum Commit hinzufügen

```
git add <file>               # Einzelne Datei zum Commit hinzufügen
git commit -m "Nachricht"    # Commit mit Nachricht
```

Status & Log

```
git status           # Zustand des Repos anzeigen
git log --oneline    # Kurze Log-Ausgabe (ein Commit pro Zeile)
```

Branches

```
git branch                # Branches anzeigen
git checkout -b feature   # Neuen Branch erstellen + wechseln
# Nachdem Feature entwickelt und committed wurde:
git checkout main         # Zurück zum main Branch
git merge feature         # Branch "feature" in main mergen
git branch -d feature     # Branch "feature" löschen
git push origin main      # Änderungen zum Remote pushen
```

Remote Repositories

```
git clone <url>                 # Repo klonen (komplette Kopie inkl. History)
git remote -v                   # Konfigurierte Remotes anzeigen (Name + URL)
git remote add origin <url>     # Neues Remote 'origin' hinzufügen
git push -u origin main         # Lokalen Branch 'main' zum Remote pushen und verknüpfen
                                # Danach reicht nur noch 'git push'
git pull --recurse-submodules   # Änderungen + referenzierte Submodule aktualisieren
```

Änderungen rückgängig machen

```
git restore <file>            # Working Directory Änderungen verwerfen
git restore --staged <file>   # Aus Stage entfernen
git revert <commit>           # Commit rückgängig (neuer Gegengift Commit)
```

Stash

```
git stash push -m "WIP"   # Änderungen im WD + Stage zwischenlagern
git stash pop             # Obersten Stash anwenden + löschen
```

```
git stash list            # Liste aller gespeicherten Stashes anzeigen
git stash clear           # Alle Stashes löschen
```

Rebase

```
git rebase main          # Rebase des aktuellen Branches auf 'main'
git rebase -i HEAD~3     # Interaktiver Rebase der letzten 3 Commits
```

Submodules

```
git submodule add <repo-url> path/to/submodule   # Submodul zu Repo hinzufügen
git clone --recurse-submodules <repo-url>        # Repo klonen und Submodule initialisieren
git pull --recurse-submodules                    # pull + referenzierte Submodule-Commits auschecken
git submodule update --remote --merge            # Submodule auf den jeweils neuesten Stand bringen
```

Subtrees

```
git subtree add --prefix=lib <repo-url> main --squash   # Fremd-Repo in lib/ einfügen
git subtree pull --prefix=lib <repo-url> main --squash  # Subtree auf neuesten Stand bringen
git subtree push --prefix=lib <repo-url> main           # Änderungen zurück ins Fremd-Repo pushen
```

.gitignore

```
echo *.png >> .gitignore    # z.B. PNG-Dateien ignorieren
```

Bisect

```
git bisect start                # Bisect-Modus starten
git bisect bad                  # Aktuellen Commit als fehlerhaft markieren
git bisect good <commit-hash>   # Bekannten funktionierenden Commit angeben
git bisect good                 # Commit funktioniert
git bisect reset                # Bisect-Modus beenden
```

Blame

```
git blame <datei>               # Zeigt, wer welche Zeile zuletzt geändert hat
git blame -L 10,20 <datei>      # Nur Zeilen 10–20 anzeigen
```

Tags

```
git tag -a v1.0 -m "Version 1.0" # Annotated Tag erstellen
git tag v1.0-light               # Lightweight Tag
git tag                          # Alle Tags anzeigen
git push origin v1.0             # Einzelnen Tag pushen
git push origin --tags           # Alle Tags pushen
```

SSH Signatur (ab Git 2.34)

```
ssh-keygen -t ed25519 -C "mail@h-da.de"       # SSH-Key erstellen
git config --global gpg.format ssh
git config --global user.signingkey ~/.ssh/id_ed25519.pub
git config --global commit.gpgsign true
```