Git

Cheat Sheet / Spickzettel

1. Git-Konfiguration (config) & Projekt anlegen (init)

lerneProgrammieren();

git config

Abrufen und definieren von Konfigurationsvariablen, die das

Aussehen und die Funktionsweise von Git steuern.

Username in Git definieren:

$ git config --global user.name "Daniel"

User-Email in Git definieren:

$ git config --global user.email "daniel@lerneprogrammieren.de"

Setze den Standard-Texteditor:

$ git config --global core.editor Vim

Eine Übersicht der Konfig-Einstellungen:

$ git config -list

git alias

Einrichten eines Alias (Abkürzung) für jeden Befehl:

$ git config --global alias.co checkout

$ git config --global alias.br branch

$ git config --global alias.ci commit

$ git config --global alias.ci commit

$ git config --global alias.st status

git init

Ein lokales Repository erstellen:

$ git init

git clone

Erstelle eine lokale Kopie des Remote-Repository.

$ git clone

2. Änderungen lokal verwalten & verfolgen (tracken)

lerneProgrammieren();

git add

Eine Datei zum Staging-Bereich hinzufügen:

$ git add <Dateiname>

Alle Dateien eines Repositories zum Staging-Bereich hinzufügen:

$ git add*

git commit

Zeichnet die Datei mit einer Meldung dauerhaft in der Versionshistorie auf oder macht einen Snapshot davon.

$ git commit -m "Meine Commit-Nachricht"

git status & show

Status des Arbeitsverzeichnisses und Staging-Bereichs.

$ git status

Zeigt Objekte an:

$ git show

git diff

Verfolge Änderungen, die nicht gestaged wurden:

$ git diff

Verfolge Änderungen, die gestaged, aber nicht comitted wurden:

$ git diff –staged

Verfolge Änderungen nach dem Commit einer Datei:

$ git diff HEAD

Verfolge Änderungen zwischen zwei Commits:

$ git diff

git diff von zwei Branches:

$ git diff <Branch 2>

3. Historie in Git, Dateien Ignorieren & entfernen

git log

Anzeige der letzten Commits sowie Status des HEAD:

$ git log

Es wird ein Commit pro Zeile angezeigt:

$ git log -oneline

Dateien anzeigen, die modifiziert wurden:

$ git log -stat

Darstellung der modifizierten Dateien mit Herkunft (Location):

$ git log -p

git blame

Änderung in jeder Zeile einer Datei anzeigen:

$ git blame <Dateiname>

.gitignore

Nicht verfolgte Dateien definieren, die Git ignorieren soll.

.gitignore-Datei erstellen:

$ touch .gitignore

Ignorierten Dateien auflisten:

$ git ls-files -i --exclude-standard

4. Branches

git branch

Branch erstellen:

$ git branch

Branches auflisten:

$ git branch --list

Branch löschen:

$ git branch -d

Remote Branch löschen

$ git push origin -delete

Branch umbenennen:

$ git branch -m

git checkout

Branch in einem Repository wechseln, ohne Commit.

Zu einem bestimmten Branch wechseln:

$ git checkout

Neuen Branch anlegen und wechseln:

$ git checkout -b

Checkout Remote Branch:

$ git checkout

git cherry pic

Änderungen eines bestehenden Commits

anwenden:

$ git cherry-pick

git stash

Branches wechseln, ohne den aktuellen Branch zu comitten

Derzeitige Arbeit (Dateien) in den Stash übertragen

$ git stash

Speichern von Stashes mit einer Nachricht:

$ git stash save „Mein Stash"

Auflisten gespeicherter Stashes:

$ git stash list

Änderungen am Stash erneut anwenden:

$ git stash apply

Stashes und deren Änderungen tracken:

$ git stash show

Vorherigen Commits erneut anwenden:

$ git stash pop

Alle verfügbaren Stashes löschen:

$ git stash clear

Stash auf einem separaten Branch:

$ git stash branch

git merge

Branches mergen (zusammenführen)

$ git merge

git rebase

Commit-Sequenz aus unterschiedlichen Branches übergeben:

$ git rebase

Rebasing-Vorgang fortsetzen:

$ git rebase -continue

Rebasing-Vorgang abbrechen:

$ git rebase --skip

git remote

Konfiguration des Remote-Servers anzeigen:

$ git remote -v

Remote hinzufügen:

$ git remote add

Daten vom Remote-Server abrufen:

$ git fetch

Remote-Verbindung entfernen:

$ git remote rm

Remote-Server umbenennen:

$ git remote rename

Zusätzliche Informationen zum Remote-Repo anzeigen:

$ git remote show

Die Remote-URL ändern:

$ git remote set-url

Daten auf den Remote-Server pushen (Übertragen):

$ git push origin master

Daten vom Remote-Server pullen (Abrufen):

$ git pull origin master

6. Remote Änderungen – Push & Pull

git push

Einen Commits vom lokalen Repository auf den Remote-Server übertragen (Pushen).

Daten zum Remote-Repository übertragen:

$ git push origin master

Push erzwingen:

$ git push -f

Remote-Branch mit dem Push-Befehl löschen:

$ git push origin -delete edited

git pull

Daten vom Server abrufen (Pull):

$ git pull origin master

Einen Remote-Branch abrufen (Pull):

$ git pull

git fetch

Download von Branches (&Tags) aus einem Repository

Ein Remote Repository abrufen (Fetch):

$ git fetch <Repository URL>

Einen bestimmten Branch abrufen:

$ git fetch

Alle Branches abrufen:

$ git fetch -all

Lokales Repository synchronisieren:

$ git fetch origin

7. Änderungen rückgängig machen (Revert & Reset)

git revert

Änderungen rückgängig machen:

$ git revert

Rückgängigmachen eines bestimmten Commits:

$ git revert

git reset (hard, soft, mixed)

Änderungen zurücksetzen (3 Modi):

$ git reset --hard

$ git reset --soft

$ git reset --mixed

git rm

Entferne Dateien aus dem Working Tree und dem Index:

$ git rm <Dateiname>

Entferne Dateien aus Git. Die Dateien bleiben im lokalen Repository bestehen:

$ git rm --cached

Jetzt noch leichter **Git lernen** mit dem LerneProgrammieren Git-Schnellstart.

Ein Online-Kurs für alle, die frustfrei Git lernen möchten.

- Ohne Vorkenntnisse
- Schritt für Schritt Anleitungen für Anfänger
- Baue echte Git Projekte

Hier klicken