Automatisiere Deine Git-Commit-Nachrichten mit ChatGPT

Das Erstellen aussagekräftiger und prägnanter Commit-Nachrichten ist ein wesentlicher Bestandteil eines guten Entwicklungsworkflows. Diese Nachrichten helfen dabei, Änderungen zu verfolgen, den Projektverlauf zu verstehen und mit Teammitgliedern zusammenzuarbeiten. Zugegeben: Das Schreiben von Commit-Nachrichten kann manchmal eine banale Aufgabe sein. In diesem Artikel zeigen ich Dir, wie Du mit ChatGPT von OpenAI automatisch Git-Commit-Nachrichten generieren lassen kannst.

Das Skript

```
#!/bin/bash

check_git_repo() {
    if ! git rev-parse --is-inside-work-tree >/dev/null 2>&1; then
        exit 1
    fi
}

check_changes() {
    if [ -z "$(git status --porcelain)" ]; then
        exit 0
    fi
}

generate_commit_message() {
    local diff_content=$(git diff --cached)
    local files_changed=$(git status --porcelain)

    echo -e "Files changed:\n$files_changed\n\nChanges:\n$diff_content" | \
        llm -m anthropic/claude-3-5-sonnet-latest \
        "Generate a git commit message for these changes. The message must
have:

        1. TITLE LINE: A specific, concise summary (max 50 chars) begin
without special characters
           that clearly describes the primary change or feature. This should
not be generic like
           'Update files' but rather describe the actual change like 'Add
user
           authentication to API endpoints'

        2. BLANK LINE

        3. DETAILED DESCRIPTION: A thorough explanation including:
```

```
            - What changes were made
            - Why they were necessary
            - Any important technical details
            - Breaking changes or important notes
            Wrap this at 72 chars.

        IMPORTANT:
        - Output ONLY the commit message
        - Make sure the title is specific to these changes
        - Focus on the what and why, not just the how"
}

# Main execution
main() {
    check_git_repo
    check_changes
    git add --all
    commit_message=$(generate_commit_message)
    git commit -m "$commit_message"
}

main "$@"
```

Aufschlüsselung

Repository-Validierung

```
check_git_repo() {
    if ! git rev-parse --is-inside-work-tree >/dev/null 2>&1; then
        exit 1
    fi
}
```

Diese Funktion stellt sicher, dass wir in einem Git-Repository arbeiten.

Änderungserkennung

```
check_changes() {
    if [ -z "$(git status --porcelain)" ]; then
        exit 0
    fi
}
```

Überprüft, ob tatsächlich Änderungen zum Festschreiben vorhanden sind.

KI-gestützte Nachrichtengenerierung

```
generate_commit_message() {
    local diff_content=$(git diff --cached)
    local files_changed=$(git status --porcelain)

    echo -e "Files changed:\n$files_changed\n\nChanges:\n$diff_content" | \
        llm -m anthropic/claude-3-5-sonnet-latest \
        "Generate a git commit message for these changes. The message must
have:

        1. TITLE LINE: A specific, concise summary (max 50 chars) that
clearly
           describes the primary change or feature. This should not be
generic like
           'Update files' but rather describe the actual change like 'Add
user
           authentication to API endpoints'

        2. BLANK LINE

        3. DETAILED DESCRIPTION: A thorough explanation including:
           - What changes were made
           - Why they were necessary
           - Any important technical details
           - Breaking changes or important notes
           Wrap this at 72 chars.

        IMPORTANT:
        - Output ONLY the commit message
        - Make sure the title is specific to these changes
        - Focus on the what and why, not just the how"
}
```

Hier geschieht die Magie – das Skript analysiert Deine Änderungen und verwendet KI, um eine aussagekräftige Commit-Nachricht zu generieren.

Das Skript verwendet Simon Willisons Kommandozeilentool llm , ein äußerst nützliches Dienstprogramm für die Interaktion mit verschiedenen KI-Modellen direkt von Deinem Terminal aus. Weitere Informationen dazu, wie Du es einrichtest und tatsächlich verwendest, findest Du in seiner Dokumentation.

Bitte beachte, dass ich in diesem Skript das Modell von Anthropic verwende. Das bedeutet, dass Du das Plugin „llm-anthropic“ einrichten musst.

Einrichten

Um das Skript am Ende auch auszuführen zu können, erstellst Du einfach eine ausführbare Commit-Datei und fügst diese in Dein Bin-Verzeichnis hinzu, sodass sie in Deinem PATH landet.

Vergiss nicht:

```
chmod +x ~/.local/bin/commit
```

um das Skript ausführbar zu machen. Ändere einfach den Pfad zum Skript, je nachdem, wo Du es speichern möchtest.

Juhuu

Nachdem Du hart an Deinem Code gearbeitet hast, musst Du ihn nur noch `**commit**` ausführen und erhälst eine von der KI generierte Commit-Nachricht.