

# AI-Powered Git Commit Messages

This python script should be placed in a directory included in your system's PATH, such as `~/bin`.

```
import subprocess
import os
from openai import OpenAI
from dotenv import load_dotenv
import argparse

load_dotenv()

def gitmessageai(push=True, only_message=False):
    # Stage all changes
    subprocess.run(["git", "add", "-A"], check=True)

    # Get the diff of staged changes
    diff_process = subprocess.run(["git", "diff", "--staged"], capture_output=True, text=True, check=True)
    diff = diff_process.stdout

    if not diff:
        print("No changes to commit.")
        return

    # Prepare the prompt for the AI
    prompt = f"""

Generate a concise commit message in Conventional Commits format for the following code changes.

Use one of the following types: feat, fix, docs, style, refactor, test, chore, perf, ci, build, or revert.

If applicable, include a scope in parentheses to describe the part of the codebase affected.

The commit message should not exceed 70 characters.

    """

    # Call the AI API to generate the commit message
    response = OpenAI().chat_completion(
        model="text-davinci-002",
        messages=[{"role": "user", "content": prompt}]

    )
```

Code changes:

{diff}

Commit message:

"""

```
# Send the prompt to the DeepSeek API
api_key = os.environ.get("DEEPSEEK_API_KEY")
if not api_key:
    print("Error: DEEPSEEK_API_KEY environment variable not set.")
    return

client = OpenAI(api_key=api_key, base_url="https://api.deepseek.com")

try:
    response = client.chat.completions.create(
        model="deepseek-chat",
        messages=[
            {"role": "user", "content": prompt}
        ],
        max_tokens=100
    )
    if response and response.choices:
        commit_message = response.choices[0].message.content.strip()
        commit_message = commit_message.replace('`', '')
    else:
        print("Error: No response from the API.")
        return
except Exception as e:
    print(f"Error during API call: {e}")
    return

# Check if the commit message is empty
if not commit_message:
    print("Error: Empty commit message generated. Aborting commit.")
```

```

    return

if only_message:
    print(f"Suggested commit message: {commit_message}")
    return

# Commit with the generated message
subprocess.run(["git", "commit", "-m", commit_message], check=True)

# Push the changes
if push:
    subprocess.run(["git", "push"], check=True)
else:
    print("Changes committed locally, but not pushed.")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Generate commit message with AI and commit changes.")
    parser.add_argument('--no-push', dest='push', action='store_false', help='Commit changes locally without pushing')
    parser.add_argument('--only-message', dest='only_message', action='store_true', help='Only print the suggested commit message')
    args = parser.parse_args()
    gitmessageai(push=args.push, only_message=args.only_message)

```

Then, in your `~/.zprofile` file, add the following:

```

alias gpa='python ~/bin/gitmessageai.py'
alias gca='python ~/bin/gitmessageai.py --no-push'
alias gm='python ~/bin/gitmessageai.py --only-message'

```