ShellGPT

LUG Frankfurt

28.01.2025

Laura Liparulo

Agenda

- Einführung in das Thema OpenAI, ChatGPT, etc.
- Prompt Engineering
- OpenAI
- Shell GPT
- Beispiele / Live Demos

OpenAI, ChatGPT, etc.

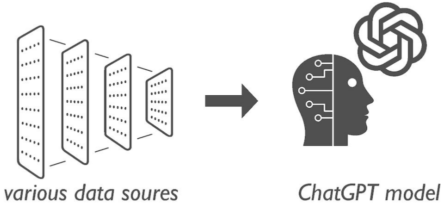
Was ist ChatGPT?

ChatGPT ist eine Gruppe großer Sprachmodelle - **large language models (LLM) -** die von OpenAI entwickelt wurde.

GPT bedeudet “ **Generative pre-training”.**

Durch das Training von Deep-Learning-Algorithmen ist das System in der Lage, auf der Grundlage vorgegebener Eingabeaufforderungen (**prompts**) menschenähnliche Textantworten zu generieren.

Es wird ein **Dialogformat** verwendet.
Wörter werden in **Token** umgewandelt.
Das System kann Wissen erwerben und weitreichende Abhängigkeiten verarbeiten, indem es mit langen Textabschnitten trainiert wird.
Die API des Chat-Modells ist zustandslos (**stateless**).



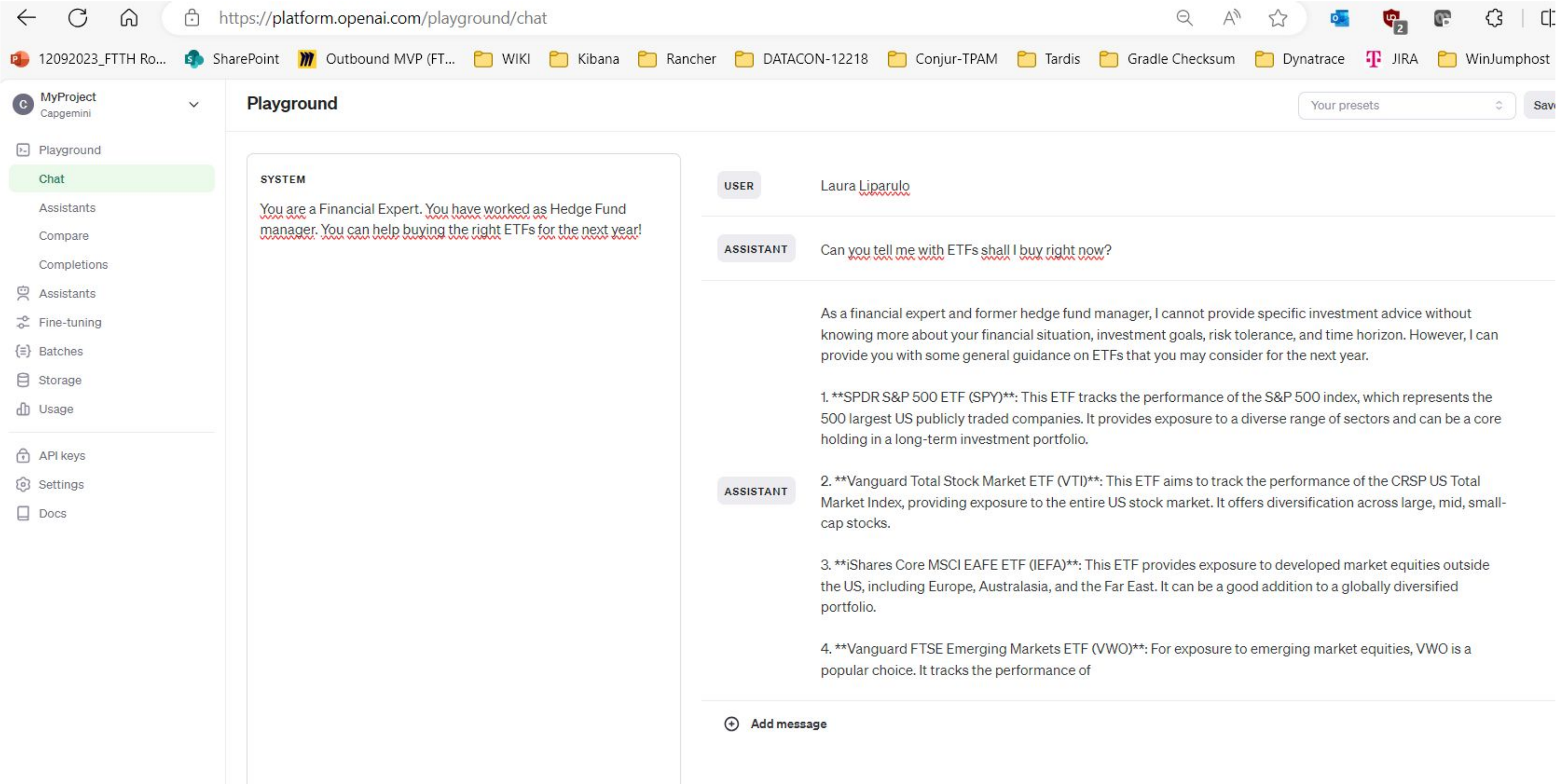
OPEN AI

OpenAI wurde 2015 von den Technologie- und Wirtschaftsführern Sam Altman, Elon Musk, Greg Brockman und Wojciech Zaremba gegründet, um sichere und offene KI-Tools zu entwickeln.

Features:

- Narrow AI - Only one task (Weak AI)
- Limited Memory AI (available as Premium users)
- Currently **pay-as-you-go** model (billing by **tokens**), ChatGPT Plus ($20 per month)
- friendly REST-API
- multimodal AI (with different types of data)
  - text (natural language)
  - visuals (images and videos)
  - audio
    - text-to-speech
    - speech-to-text
    - sound
    - music

OpenAI - GUI - Ex. Financial Expert



SYSTEM

You are a Financial Expert. You have worked as Hedge Fund manager. You can help buying the right ETFs for the next year!

USER

Laura Liparulo

ASSISTANT

Can you tell me with ETFs shall I buy right now?

ASSISTANT

As a financial expert and former hedge fund manager, I cannot provide specific investment advice without knowing more about your financial situation, investment goals, risk tolerance, and time horizon. However, I can provide you with some general guidance on ETFs that you may consider for the next year.

1. **SPDR S&P 500 ETF (SPY)**: This ETF tracks the performance of the S&P 500 index, which represents the 500 largest US publicly traded companies. It provides exposure to a diverse range of sectors and can be a core holding in a long-term investment portfolio.

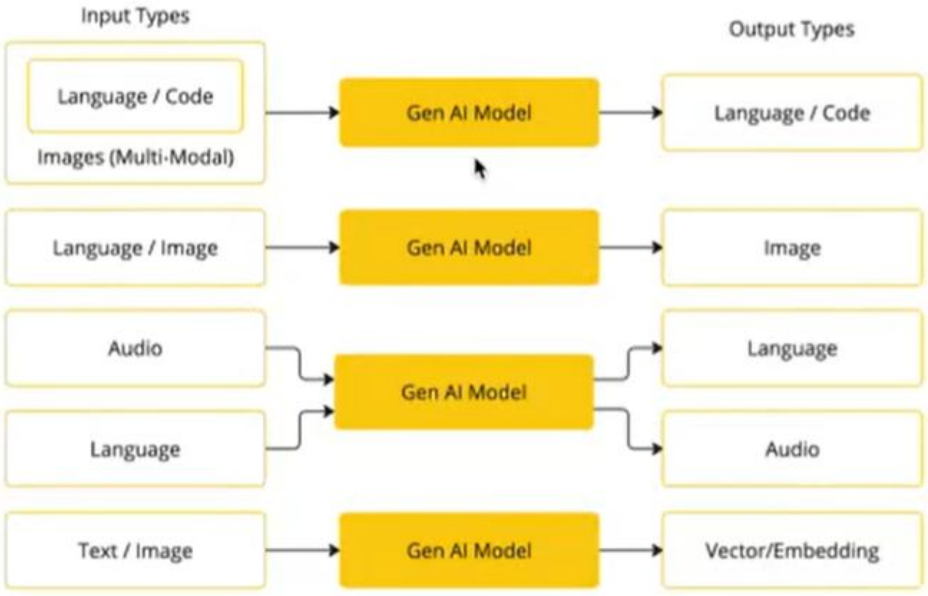
2. **Vanguard Total Stock Market ETF (VTI)**: This ETF aims to track the performance of the CRSP US Total Market Index, providing exposure to the entire US stock market. It offers diversification across large, mid, small-cap stocks.

3. **iShares Core MSCI EAFE ETF (IEFA)**: This ETF provides exposure to developed market equities outside the US, including Europe, Australasia, and the Far East. It can be a good addition to a globally diversified portfolio.

4. **Vanguard FTSE Emerging Markets ETF (VWO)**: For exposure to emerging market equities, VWO is a popular choice. It tracks the performance of

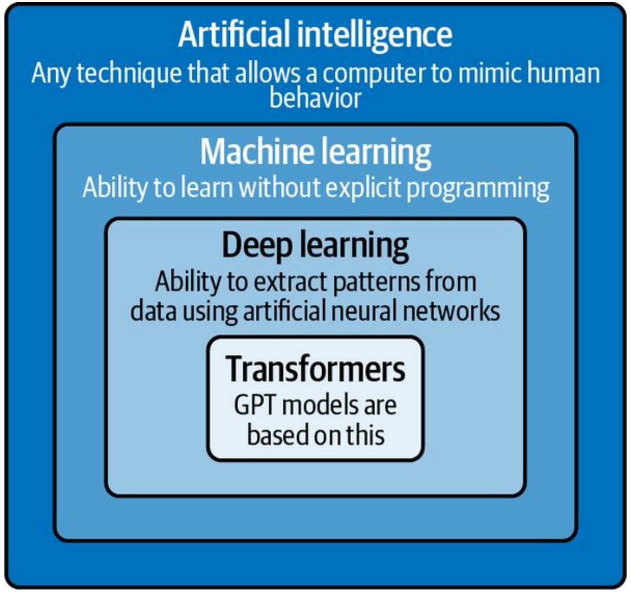
Generative AI

Generative KI (GenAI) ist eine Art von KI, die sich auf die Generierung von Inhalten wie Text, Bildern, Video oder Audio konzentriert.



AI - ML - Deep Learning - NLP

- ***Artificial Intelligence*** - Entwicklung von Systemen kann Aufgaben ausführen, die typischerweise menschliche Intelligenz erfordern
- ***Machine Learning*** - eine Teilmenge der KI – Algorithmen, die es dem System ermöglichen, selbst zu lernen
- ***Deep learning*** (***neural networks)***- ein Zweig des ML, der sich auf Algorithmen konzentriert, die von der Struktur des Gehirns inspiriert sind-
  - image and speech recognition
  - natural language processing (NLP).
- ***NLP*** - Teilgebiet der KI, der sich darauf konzentriert, Computer in die Lage zu versetzen, natürliche menschliche Sprache zu verarbeiten, zu interpretieren und zu erzeugen



Artificial intelligence

Any technique that allows a computer to mimic human behavior

Machine learning

Ability to learn without explicit programming

Deep learning

Ability to extract patterns from data using artificial neural networks

Transformers

GPT models are based on this

Transformers

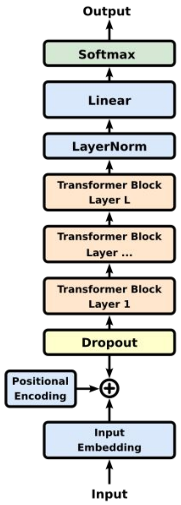
Algorithmen, die auf einem Aufmerksamkeitsmechanismus (“**Attention mechanism**”) basieren und ein tieferes Verständnis des Kontexts ermöglichen und kohärente Ergebnisse ermöglichen.

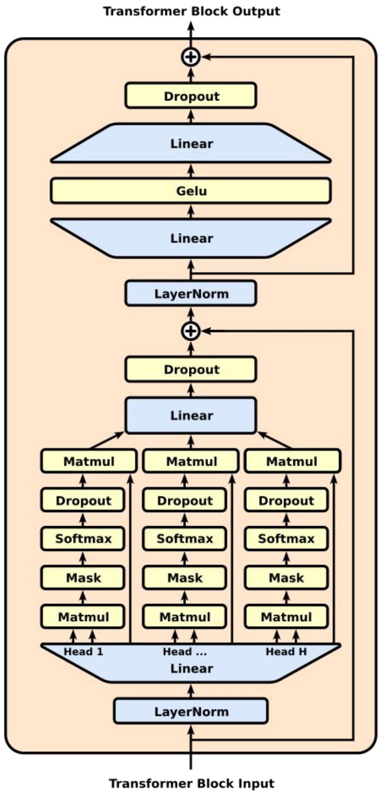
Es handelt sich um **Reinforcement learning**.

Parallelität und **Skalierbarkeit** sind vorausgesetzt, um außergewöhnlich große KI-Modelle und deren Trainingsanforderungen zu bewältigen.

Das Verständnis der Bedeutung von Wörtern innerhalb von Sätzen ermöglicht die Erzeugung von:

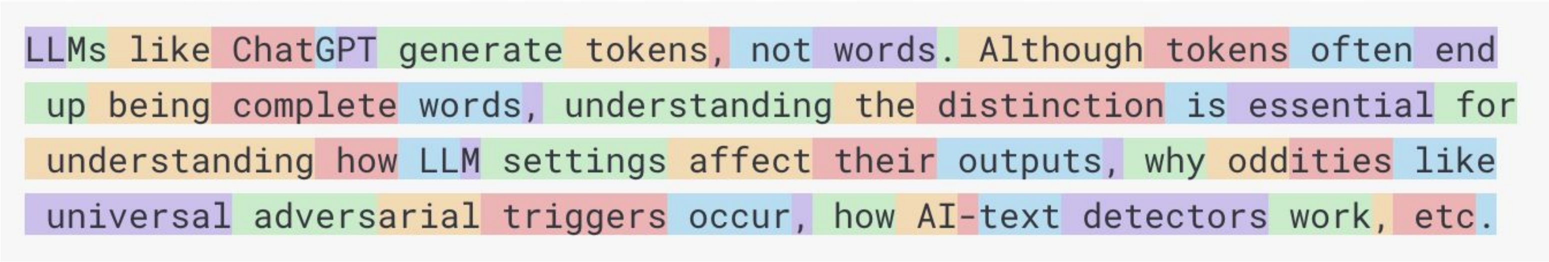
- Übersetzung
- Beantwortung von Fragen
- Texterstellung
- etc.


Output
Softmax
Linear
LayerNorm
Transformer Block Layer L
Transformer Block Layer ...
Transformer Block Layer 1
Dropout
Positional Encoding
Input Embedding
Input



Tokens

- **Eingabe (input)**: Modelle wandeln Wörter in Token um
- **Ausgabe (output)**: Token werden wieder in Wörter umgewandelt
- Sowohl die Eingabe als auch die Ausgabe tragen zur Gesamtzahl der Token bei, die die Abrechnung ausmacht
- Im Englischen entspricht ein Token ungefähr 75 % eines Wortes.
- Shakespeares Gesamtwerk mit einem Gesamtumfang von rund 900.000 Wörtern ergibt etwa 1,2 Millionen Zeichen.



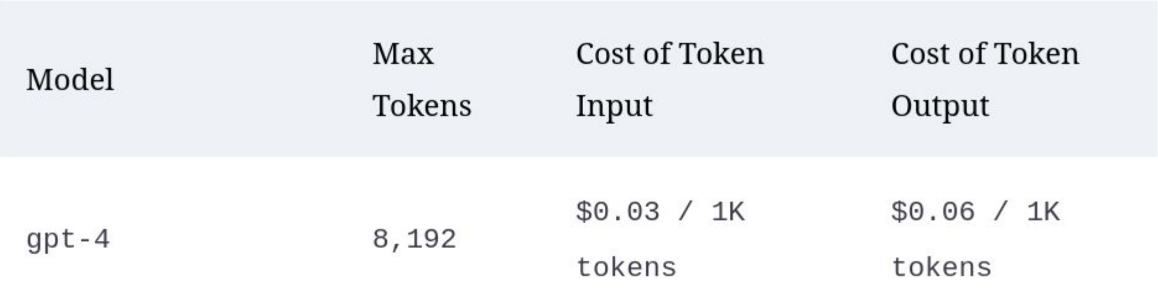
LLMs like ChatGPT generate tokens, not words. Although tokens often end up being complete words, understanding the distinction is essential for understanding how LLM settings affect their outputs, why oddities like universal adversarial triggers occur, how AI-text detectors work, etc.

GPT evolution

- **ELIZA (1966) -** Einfache erste Sprachmodelle
- **BERT (2017) -** Training für große Modelle
- **GPT (2018)**
- **GPT-2 (2019)** - Mit deutlich mehr Parametern als sein Vorgänger
- **GPT-3 (2020)**
  - Ein Sprung in Größe und Leistungsfähigkeit
  - 175 Milliarden Parameter ermöglichten ein beispielloses Maß an Verständnis und Erzeugung näturlicher Sprache
- **GPT-4 (2023)**
  - GPT-4 (geschätzte 1,76 Billionen Parameter) führte multimodale Funktionen ein, verarbeitete sowohl Text- als auch Bildeingaben und zeigte eine überlegene Leistung bei komplexen Aufgaben, einschließlich des Bestehens einer simulierten Anwaltsprüfung mit Ergebnissen in den besten 10 %.
    - **GPT-4 Vision -** significant advancement as a multimodal model, allowing it to process not only text but also images as input
    - **GPT-4 Turbo** t can process and generate text, vision, and audio
    - **GPT-4o -** with a larger input context window of 128,000 tokens, equivalent to almost three hundred pages of English text..

Token limits

- Einschränkungen hinsichtlich der Textmenge, die in einem einzelnen API-Aufruf verarbeitet wird.
- Der Schwellenwert wird oft als „Kontextfenster“ bezeichnet.
- Das Modell verarbeitet keinen Text, der diesen Grenzwert überschreitet.
- ChatGPT3 hat ein 4K-Token-Limit
- GPT4 bietet verschiedene Optionen, wie 8K, 16K und 32K.



Model	Max Tokens	Cost of Token Input	Cost of Token Output
`gpt-4`	`8,192`	`$0.03 / 1K tokens`	`$0.06 / 1K tokens`

Andere Open AI models

- Fortschritte in der KI-generierten Kunst und Bildgebung
    - **DALL·E** (2021):
    - **DALL·E 2** and **3** (2022-2023):
    - **Sora** - for videos

- Codierung, Untertitel und Sprache
    - **Codex** - Aufbauend auf GPT-3 wurde Codex auf Milliarden von Codezeilen trainiert, um die Softwareentwicklung zu unterstützen und Tools wie GitHub Copilot zu unterstützen – nicht mehr öffentlich
    - **Clip** - um präzise Bildunterschriften für eine Vielzahl von Bildern zu erstellen
    - **Whisper** - Ein fortschrittliches automatisches Spracherkennungssystem (maximal 25 MB pro Datei)
    - **Moderation**

Wie LLMs funktionieren

Der Prozess ist in 3 Schritte unterteilt:

1. Aufteilen von Text in Token (Wortstücke, Tokenisierung)
2. Konvertieren von Token in numerische Vektoren (Einbettung)
3. Vektordatenbanken zum Speichern/Abrufen von Worteinbettungen und -beziehungen

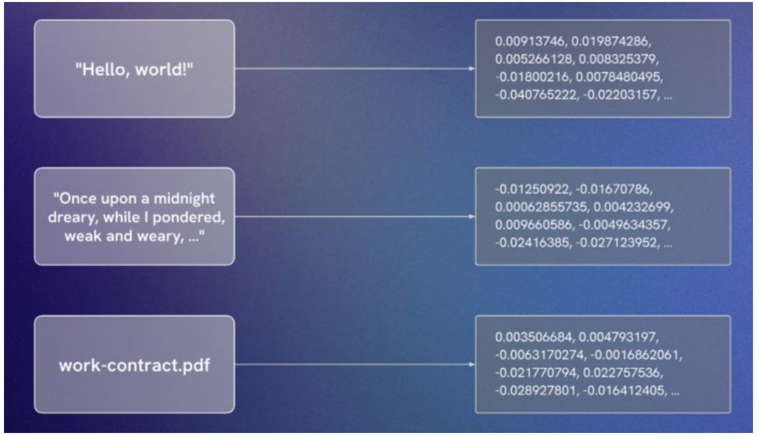
Prognostizieren Sie das nächste Wort basierend auf dem Kontext aus Trainingsdaten mithilfe von Transformern – Algorithmen wie der Multi-Head-Aufmerksamkeit auf Prozessvektoren

Fine-Tuning :

- Weitere Schulung großer Basismodelle für spezifische Anwendungsfälle
- Ermöglicht die Anpassung bei gleichzeitiger Nutzung vorab trainierter Funktionen

Embedding - Vector DBs

Texte und Dokumente können in Vektoren zerlegt und in Vektordatenbanken gespeichert werden



chroma
Pinecone
supabase

Features:

- Helfen Sie dabei, API-Aufrufe und die Token-Nutzung zu reduzieren
- Chatverläufe zusammenfassen und speichern
- enorme Reduzierung der Halluzinationen

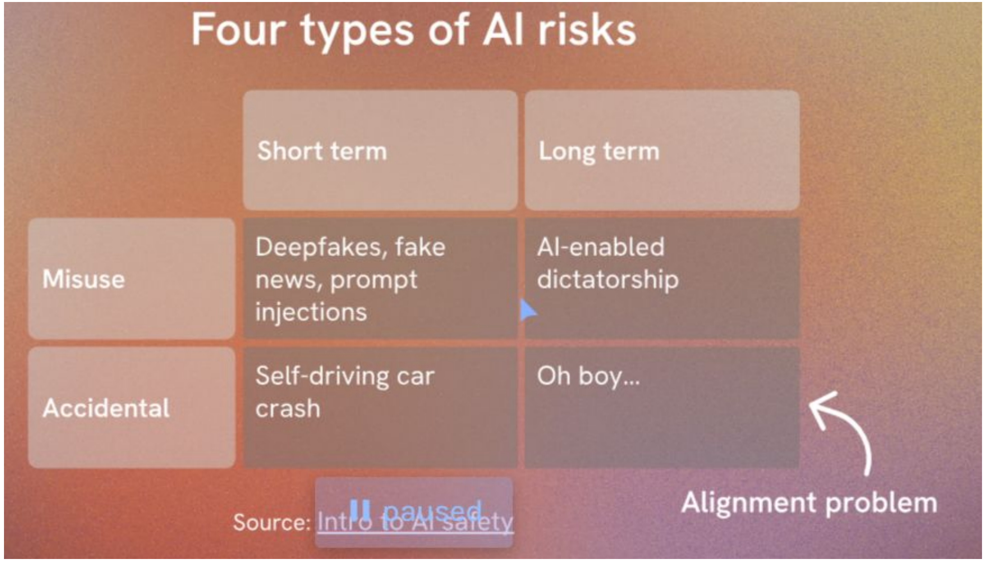
Einschränkungen und Herausforderungen der LLMs

- by design nicht zuverlässig
- **Bias**, Halluzinationen, mathematical reasoning weaknesses (Ungenauigkeiten, inkohärente Antworten)
- Begrenzter Wissensabbau (knowledge-cutoff), der durch Surfen im Internet gemildert wird
- Beschränkt auf die letzte Aktualisierung der Trainingsdaten (2023)
- Unfähigkeit, Fakten zu überprüfen
- Hoher Rechenaufwand
- mögliche Urheberrechtsverletzungen
- Möglicher Missbrauch für Fehlinformationen, der die Belegschaft stört

Plug-ins als Lösung (z.B. Medizin)

Risks of GenAI

- **Prompt injecting** - eine neue Angriffstechnik, die es Angreifern ermöglicht, die Ausgabe des LLM zu manipulieren
- **Short Term vs Long Term**



Four types of AI risks

	Short term	Long term
Misuse	Deepfakes, fake news, prompt injections	AI-enabled dictatorship
Accidental	Self-driving car crash	Oh boy...

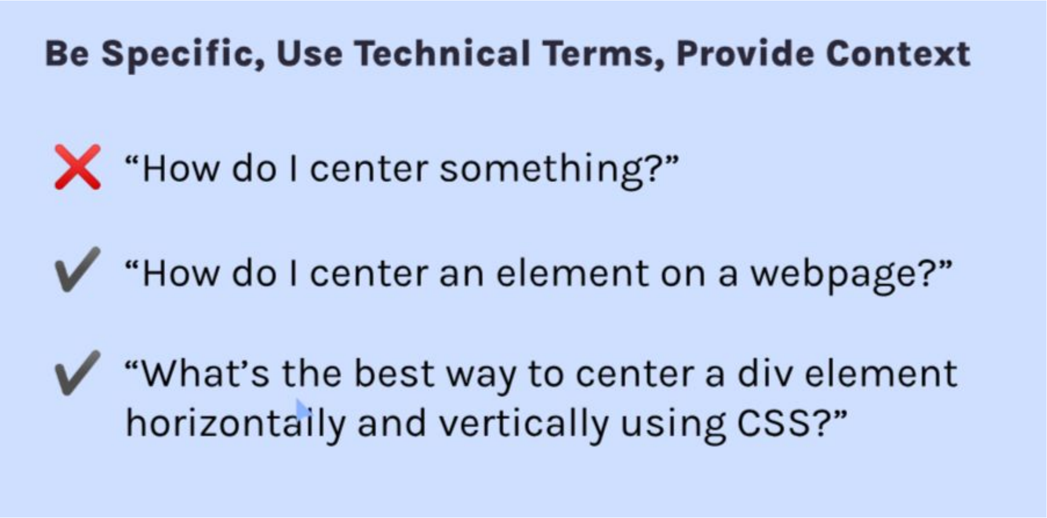
Alignment problem

Source: Intro to AI safety

Prompt Engineering

Prompt engineering

- “the art of getting the AI know what to do”



Be Specific, Use Technical Terms, Provide Context

❌ “How do I center something?”

✔ “How do I center an element on a webpage?”

✔ “What’s the best way to center a div element horizontally and vertically using CSS?”

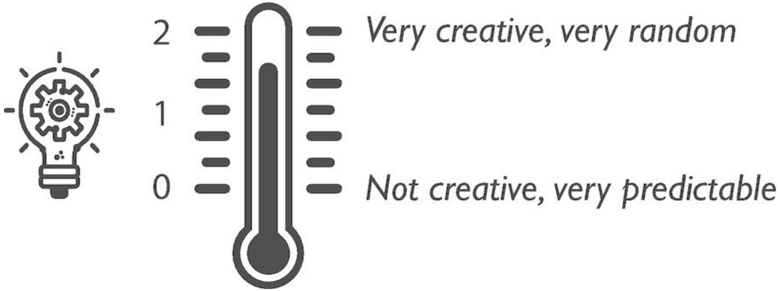
Prompt engineering techniques

- **Prompt Design** - klare, gut strukturierte Aufforderungen zur Kommunikation der Aufgaben
- **Augmentation** - Hinzufügen von relevantem Kontext, Beispielen, Einschränkungen usw.
- . (RAG - Vector databases)
- **Chaining** - Aufteilen komplexer Aufgaben in eine Reihe kleinerer Unteraufforderungen
- **Prompt tuning** - Anpassen des Modells an neue nachgelagerte Aufgaben, ohne das Modell neu zu trainieren und seine Gewichtung zu aktualisieren

Temperature

Indem Sie die Temperatureinstellung in Ihren Eingabeaufforderungen an die ChatGPT-API anpassen, können Sie die Antworten kreativer gestalten.

Der Temperaturwert reicht von **0 bis 2**, wobei niedrigere Werte auf einen größeren Determinismus und höhere Werte auf mehr Zufälligkeit hinweisen.



ShellGPT

Was is ShellGPT?

- Command-line productivity tool, angetrieben durch OpenAI LLMs (GPT-4)
- Linux, macOS, Windows unterstützt
- kompatibel mit allen wichtigen Shells wie PowerShell, CMD, Bash, Zsh, etc.
- im Python geschrieben
- Started by Farkhod Sadykov, derzeit gibt es 33 contributors
- kann in Azure und Ollama integriert werden
- https://github.com/TheR1D/shell_gpt.git

Installing ShellGPT

It requires Python

```
> pip3 install shell-gpt
```

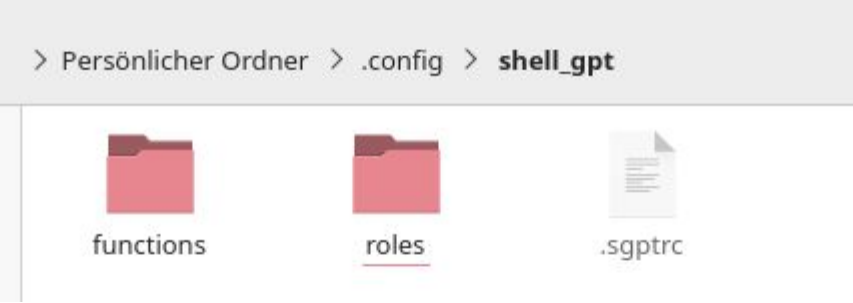
You need to define the **OPEN_AI_API_KEY**:

```
> export OPENAI_API_KEY=<your secret API key>
```

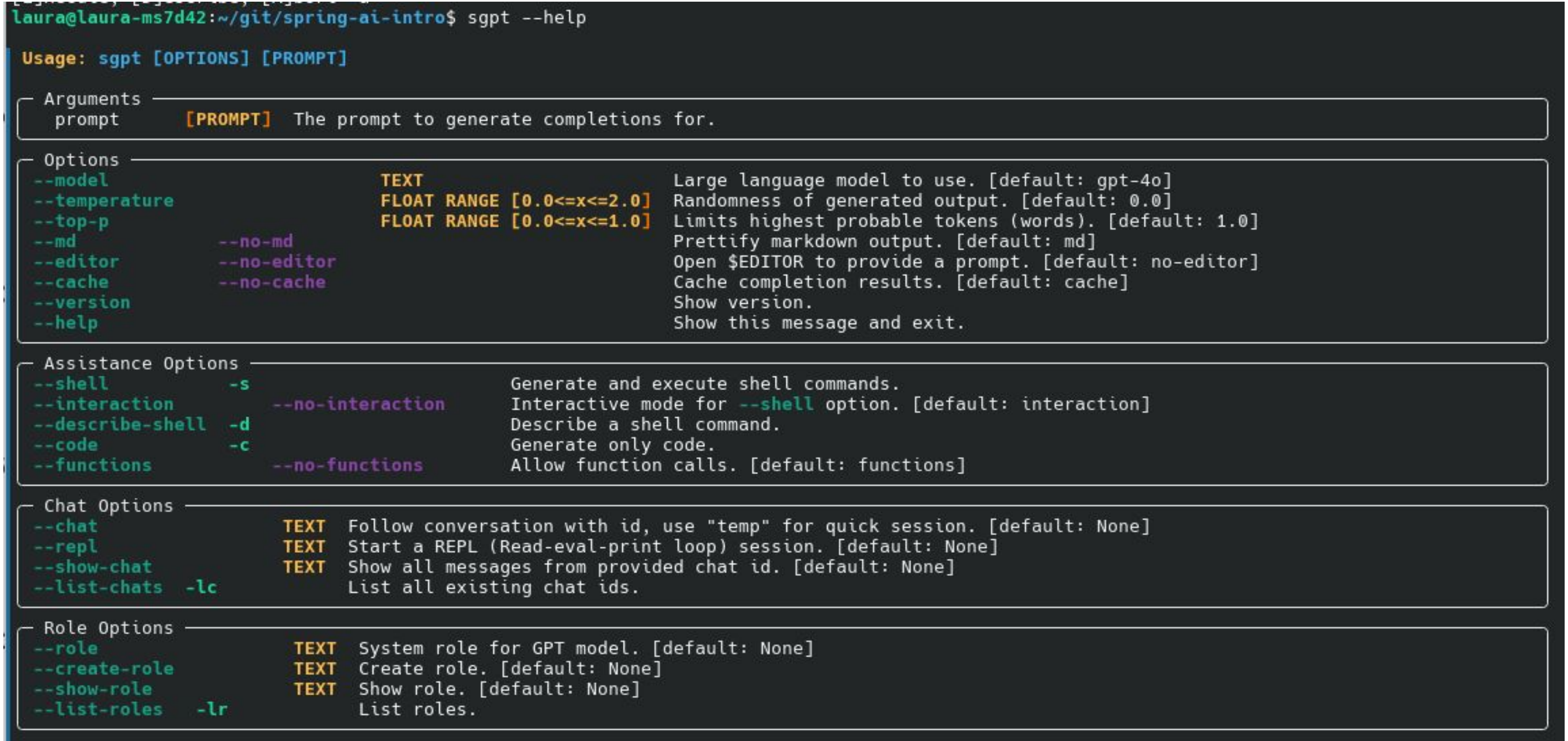
Configuring Shell GPT

You can setup some parameters in runtime configuration file **`~/.config/shell_gpt/.sgptrc`**:

```
# API key, also it is possible to define OPENAI_API_KEY env.
OPENAI_API_KEY=your_api_key
# Base URL of the backend server. If "default" URL will be resolved based on --model.
API_BASE_URL=default
# Max amount of cached message per chat session.
CHAT_CACHE_LENGTH=100
# Chat cache folder.
CHAT_CACHE_PATH=/tmp/shell_gpt/chat_cache
# Request cache length (amount).
CACHE_LENGTH=100
# Request cache folder.
CACHE_PATH=/tmp/shell_gpt/cache
# Request timeout in seconds.
REQUEST_TIMEOUT=60
# Default OpenAI model to use.
DEFAULT_MODEL=gpt-4o
# Default color for shell and code completions.
DEFAULT_COLOR=magenta
# When in --shell mode, default to "Y" for no input.
DEFAULT_EXECUTE_SHELL_CMD=false
# Disable streaming of responses
DISABLE_STREAMING=false
# The pygment theme to view markdown (default/describe role).
CODE_THEME=default
# Path to a directory with functions.
OPENAI_FUNCTIONS_PATH=/Users/user/.config/shell_gpt/functions
# Print output of functions when LLM uses them.
SHOW_FUNCTIONS_OUTPUT=false
# Allows LLM to use functions.
OPENAI_USE_FUNCTIONS=true
# Enforce LiteLLM usage (for local LLMs).
USE_LITELLM=false
```

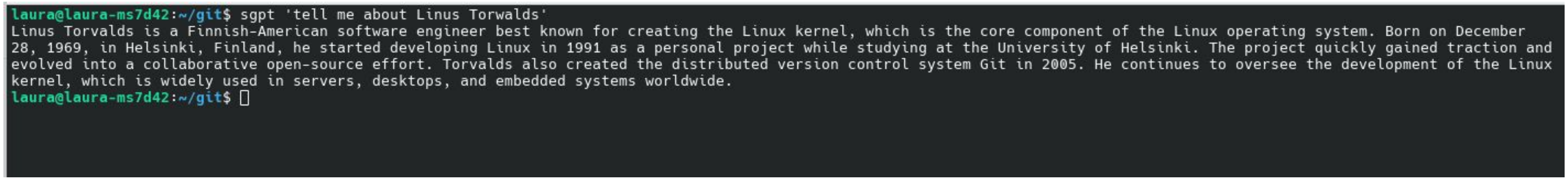


Shell GPT Help



```
laura@laura-ms7d42:~/git/spring-ai-intro$ sgpt --help

 Usage: sgpt [OPTIONS] [PROMPT]

╭─ Arguments ──────────────────────────────────────────────────────────────────────────────────────────────╮
│  prompt      [PROMPT]  The prompt to generate completions for.                                           │
╰──────────────────────────────────────────────────────────────────────────────────────────────────────────╯
╭─ Options ────────────────────────────────────────────────────────────────────────────────────────────────╮
│ --model                            TEXT                       Large language model to use. [default: gpt-4o]   │
│ --temperature                      FLOAT RANGE [0.0<=x<=2.0]  Randomness of generated output. [default: 0.0]   │
│ --top-p                            FLOAT RANGE [0.0<=x<=1.0]  Limits highest probable tokens (words). [default: 1.0] │
│ --md               --no-md                                    Prettify markdown output. [default: md]          │
│ --editor           --no-editor                                Open $EDITOR to provide a prompt. [default: no-editor] │
│ --cache            --no-cache                                 Cache completion results. [default: cache]       │
│ --version                                                     Show version.                                    │
│ --help                                                        Show this message and exit.                      │
╰──────────────────────────────────────────────────────────────────────────────────────────────────────────╯
╭─ Assistance Options ─────────────────────────────────────────────────────────────────────────────────────╮
│ --shell           -s                         Generate and execute shell commands.                        │
│ --interaction         --no-interaction       Interactive mode for --shell option. [default: interaction] │
│ --describe-shell  -d                         Describe a shell command.                                   │
│ --code            -c                         Generate only code.                                         │
│ --functions           --no-functions         Allow function calls. [default: functions]                  │
╰──────────────────────────────────────────────────────────────────────────────────────────────────────────╯
╭─ Chat Options ───────────────────────────────────────────────────────────────────────────────────────────╮
│ --chat                  TEXT  Follow conversation with id, use "temp" for quick session. [default: None] │
│ --repl                  TEXT  Start a REPL (Read-eval-print loop) session. [default: None]               │
│ --show-chat             TEXT  Show all messages from provided chat id. [default: None]                   │
│ --list-chats  -lc             List all existing chat ids.                                                │
╰──────────────────────────────────────────────────────────────────────────────────────────────────────────╯
╭─ Role Options ───────────────────────────────────────────────────────────────────────────────────────────╮
│ --role                   TEXT  System role for GPT model. [default: None]                                │
│ --create-role            TEXT  Create role. [default: None]                                              │
│ --show-role              TEXT  Show role. [default: None]                                                │
│ --list-roles   -lr             List roles.                                                               │
╰──────────────────────────────────────────────────────────────────────────────────────────────────────────╯
```

Example - Simple question



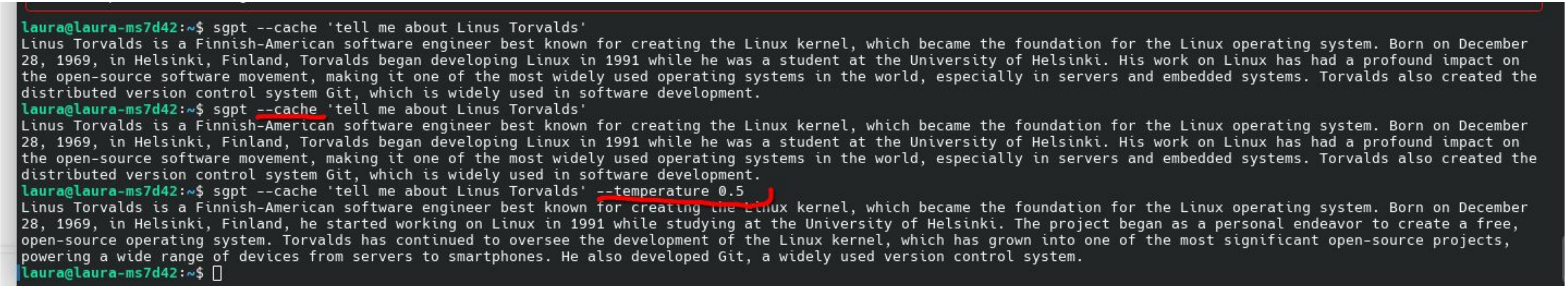
```
laura@laura-ms7d42:~/git$ sgpt 'tell me about Linus Torwalds'
Linus Torvalds is a Finnish-American software engineer best known for creating the Linux kernel, which is the core component of the Linux operating system. Born on December
28, 1969, in Helsinki, Finland, he started developing Linux in 1991 as a personal project while studying at the University of Helsinki. The project quickly gained traction and
evolved into a collaborative open-source effort. Torvalds also created the distributed version control system Git in 2005. He continues to oversee the development of the Linux
kernel, which is widely used in servers, desktops, and embedded systems worldwide.
laura@laura-ms7d42:~/git$ 
```

Request cache

Control cache using `--cache` (default) and `--no-cache` options. This caching applies for all `sgpt` requests to OpenAI API

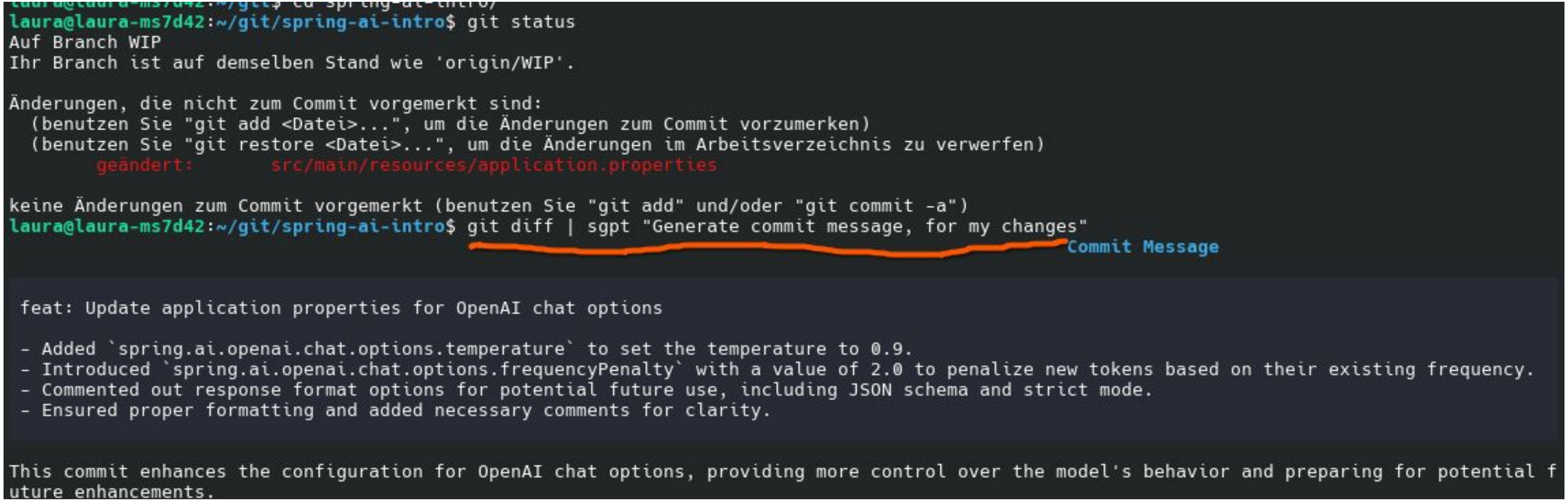
If you add the temperature, the answer will change.

Example:



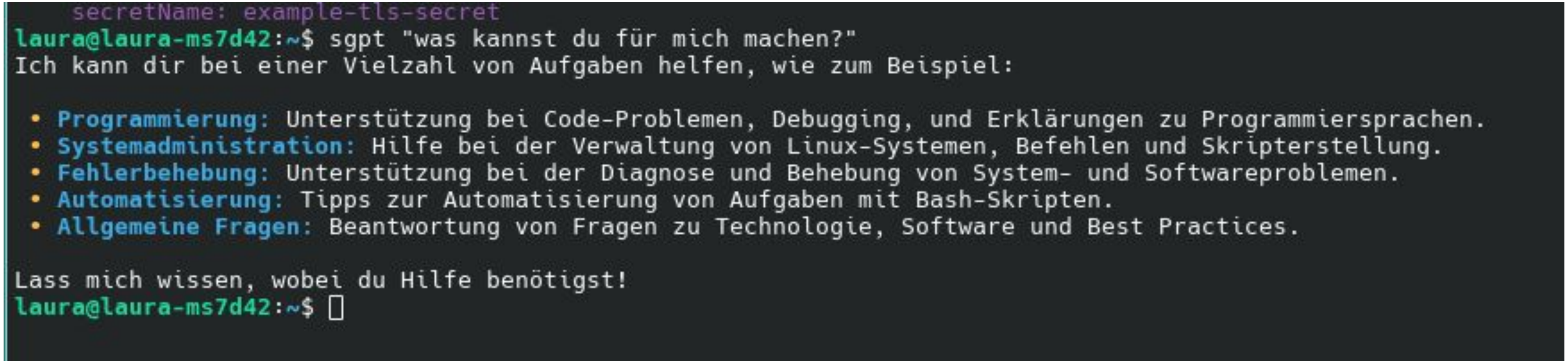
```
laura@laura-ms7d42:~$ sgpt --cache 'tell me about Linus Torvalds'
Linus Torvalds is a Finnish-American software engineer best known for creating the Linux kernel, which became the foundation for the Linux operating system. Born on December
28, 1969, in Helsinki, Finland, Torvalds began developing Linux in 1991 while he was a student at the University of Helsinki. His work on Linux has had a profound impact on
the open-source software movement, making it one of the most widely used operating systems in the world, especially in servers and embedded systems. Torvalds also created the
distributed version control system Git, which is widely used in software development.
laura@laura-ms7d42:~$ sgpt --cache 'tell me about Linus Torvalds'
Linus Torvalds is a Finnish-American software engineer best known for creating the Linux kernel, which became the foundation for the Linux operating system. Born on December
28, 1969, in Helsinki, Finland, Torvalds began developing Linux in 1991 while he was a student at the University of Helsinki. His work on Linux has had a profound impact on
the open-source software movement, making it one of the most widely used operating systems in the world, especially in servers and embedded systems. Torvalds also created the
distributed version control system Git, which is widely used in software development.
laura@laura-ms7d42:~$ sgpt --cache 'tell me about Linus Torvalds' --temperature 0.5
Linus Torvalds is a Finnish-American software engineer best known for creating the Linux kernel, which became the foundation for the Linux operating system. Born on December
28, 1969, in Helsinki, Finland, he started working on Linux in 1991 while studying at the University of Helsinki. The project began as a personal endeavor to create a free,
open-source operating system. Torvalds has continued to oversee the development of the Linux kernel, which has grown into one of the most significant open-source projects,
powering a wide range of devices from servers to smartphones. He also developed Git, a widely used version control system.
laura@laura-ms7d42:~$ 
```

Example - Git commit message



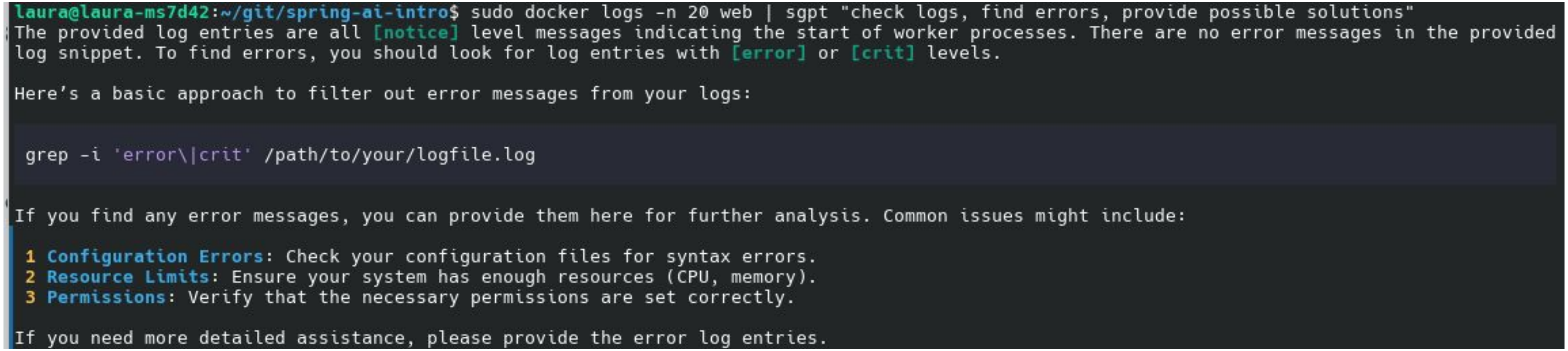
```
laura@laura-ms7d42:~/git$ cd spring-ai-intro/
laura@laura-ms7d42:~/git/spring-ai-intro$ git status
Auf Branch WIP
Ihr Branch ist auf demselben Stand wie 'origin/WIP'.

Änderungen, die nicht zum Commit vorgemerkt sind:
  (benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
  (benutzen Sie "git restore <Datei>...", um die Änderungen im Arbeitsverzeichnis zu verwerfen)
        geändert:       src/main/resources/application.properties

keine Änderungen zum Commit vorgemerkt (benutzen Sie "git add" und/oder "git commit -a")
laura@laura-ms7d42:~/git/spring-ai-intro$ git diff | sgpt "Generate commit message, for my changes"
                                                                                                     Commit Message

 feat: Update application properties for OpenAI chat options

 - Added `spring.ai.openai.chat.options.temperature` to set the temperature to 0.9.
 - Introduced `spring.ai.openai.chat.options.frequencyPenalty` with a value of 2.0 to penalize new tokens based on their existing frequency.
 - Commented out response format options for potential future use, including JSON schema and strict mode.
 - Ensured proper formatting and added necessary comments for clarity.

This commit enhances the configuration for OpenAI chat options, providing more control over the model's behavior and preparing for potential f
uture enhancements.
```

Example - Language recognition



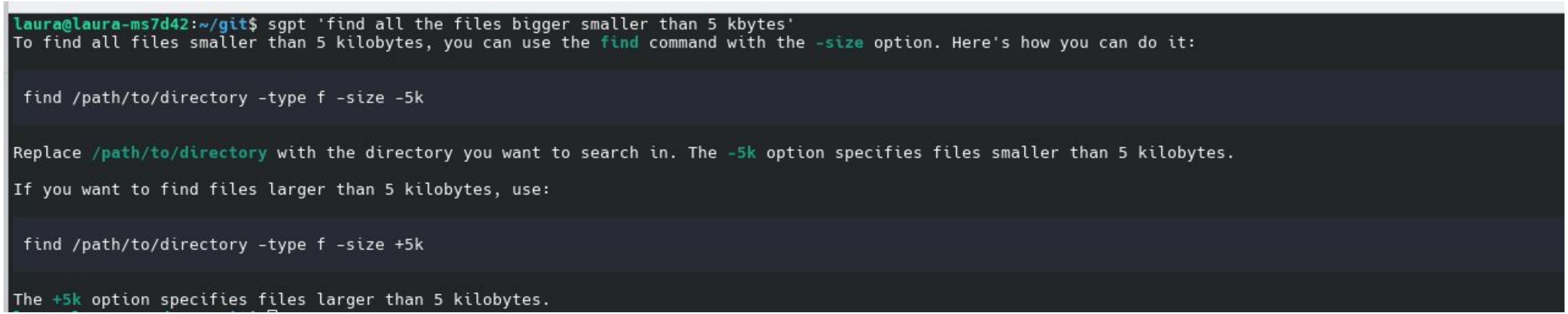
```
    secretName: example-tls-secret
laura@laura-ms7d42:~$ sgpt "was kannst du für mich machen?"
Ich kann dir bei einer Vielzahl von Aufgaben helfen, wie zum Beispiel:

 • Programmierung: Unterstützung bei Code-Problemen, Debugging, und Erklärungen zu Programmiersprachen.
 • Systemadministration: Hilfe bei der Verwaltung von Linux-Systemen, Befehlen und Skripterstellung.
 • Fehlerbehebung: Unterstützung bei der Diagnose und Behebung von System- und Softwareproblemen.
 • Automatisierung: Tipps zur Automatisierung von Aufgaben mit Bash-Skripten.
 • Allgemeine Fragen: Beantwortung von Fragen zu Technologie, Software und Best Practices.

Lass mich wissen, wobei du Hilfe benötigst!
laura@laura-ms7d42:~$ 
```

Example - Docker logs analyzing



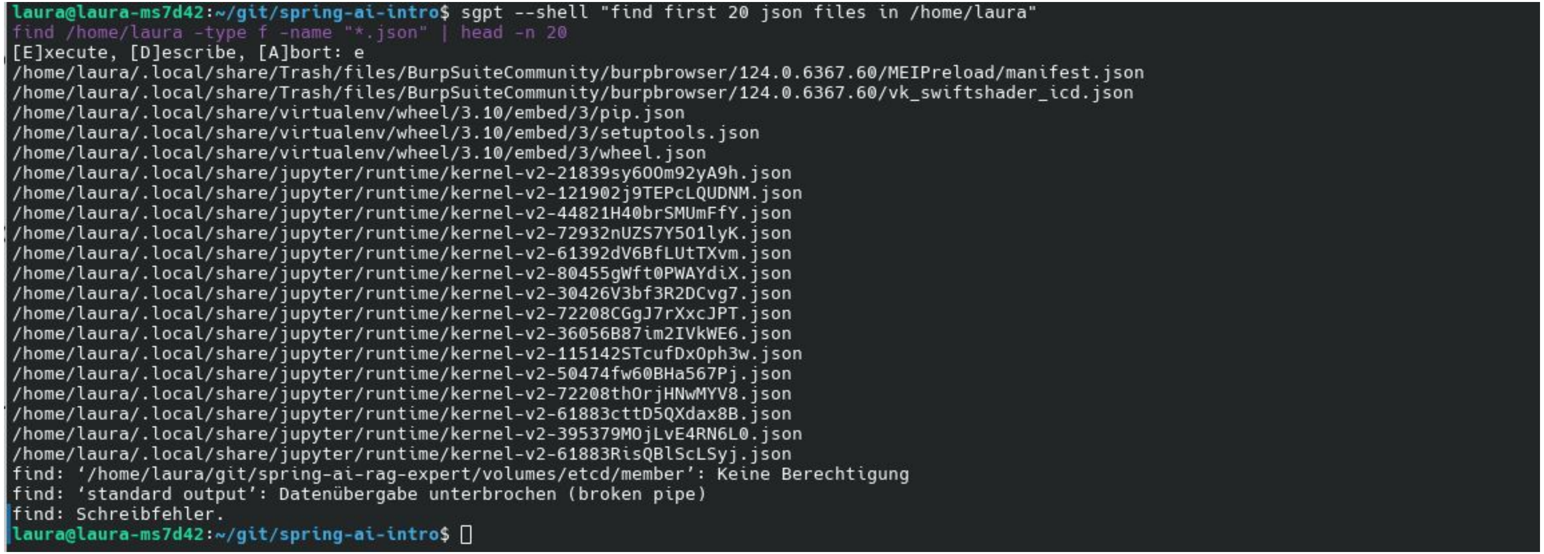
```
laura@laura-ms7d42:~/git/spring-ai-intro$ sudo docker logs -n 20 web | sgpt "check logs, find errors, provide possible solutions"
The provided log entries are all [notice] level messages indicating the start of worker processes. There are no error messages in the provided
log snippet. To find errors, you should look for log entries with [error] or [crit] levels.

Here's a basic approach to filter out error messages from your logs:

 grep -i 'error\|crit' /path/to/your/logfile.log

If you find any error messages, you can provide them here for further analysis. Common issues might include:

 1 Configuration Errors: Check your configuration files for syntax errors.
 2 Resource Limits: Ensure your system has enough resources (CPU, memory).
 3 Permissions: Verify that the necessary permissions are set correctly.

If you need more detailed assistance, please provide the error log entries.
```

Example 1 - Shell GPT for Linux commands



```
laura@laura-ms7d42:~/git$ sgpt 'find all the files bigger smaller than 5 kbytes'
To find all files smaller than 5 kilobytes, you can use the find command with the -size option. Here's how you can do it:

 find /path/to/directory -type f -size -5k

Replace /path/to/directory with the directory you want to search in. The -5k option specifies files smaller than 5 kilobytes.

If you want to find files larger than 5 kilobytes, use:

 find /path/to/directory -type f -size +5k

The +5k option specifies files larger than 5 kilobytes.
```

Shell commands generator with “--shell” or “-s” option

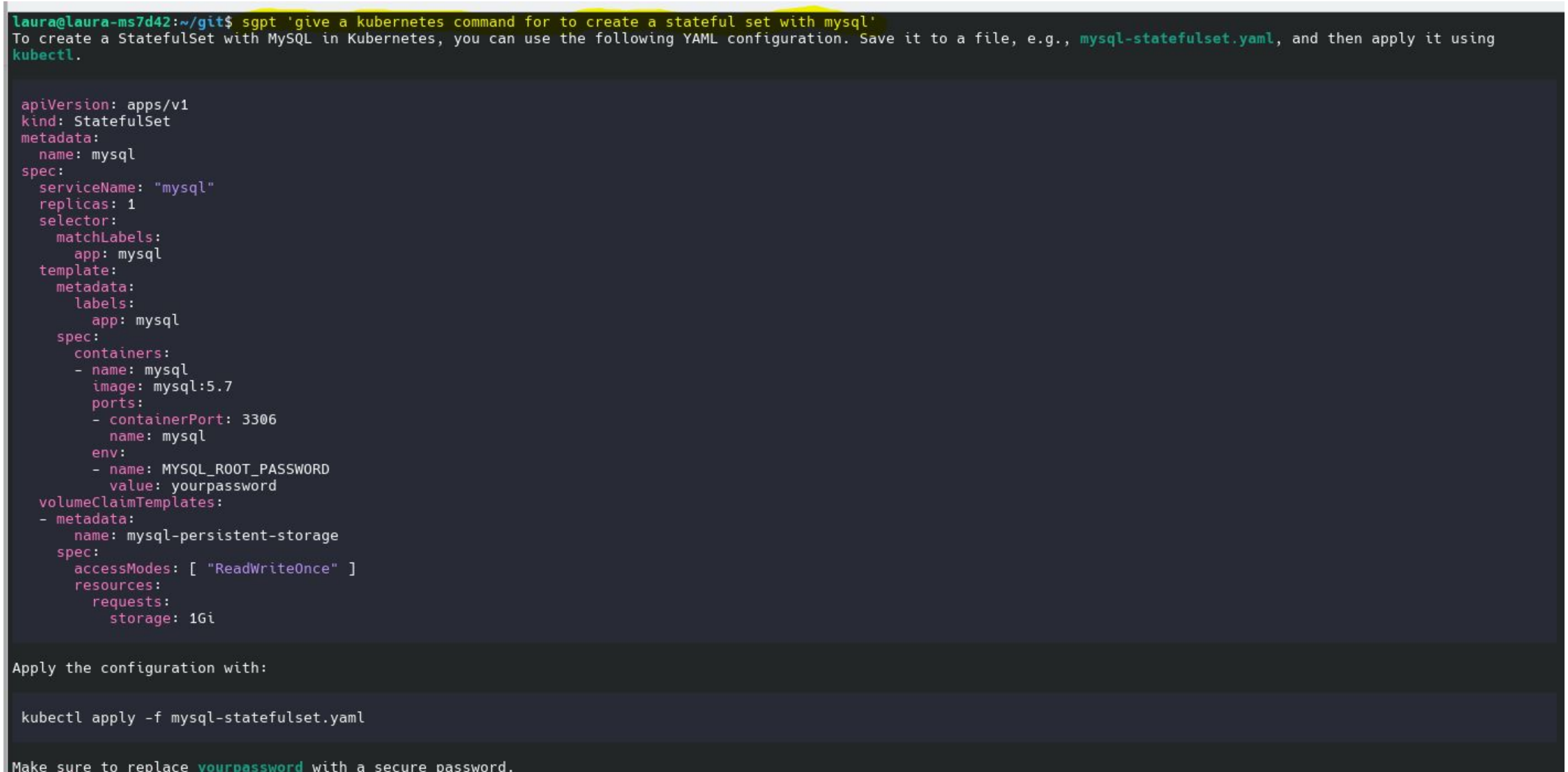


```
laura@laura-ms7d42:~/git/spring-ai-intro$ sgpt --shell "find first 20 json files in /home/laura"
find /home/laura -type f -name "*.json" | head -n 20
[E]xecute, [D]escribe, [A]bort: e
/home/laura/.local/share/Trash/files/BurpSuiteCommunity/burpbrowser/124.0.6367.60/MEIPreload/manifest.json
/home/laura/.local/share/Trash/files/BurpSuiteCommunity/burpbrowser/124.0.6367.60/vk_swiftshader_icd.json
/home/laura/.local/share/virtualenv/wheel/3.10/embed/3/pip.json
/home/laura/.local/share/virtualenv/wheel/3.10/embed/3/setuptools.json
/home/laura/.local/share/virtualenv/wheel/3.10/embed/3/wheel.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-21839sy6OOm92yA9h.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-121902j9TEPcLQUDNM.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-44821H40brSMUmFfY.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-72932nUZS7Y5O1lyK.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-61392dV6BfLUtTXvm.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-80455gWft0PWAYdiX.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-30426V3bf3R2DCvg7.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-72208CGgJ7rXxcJPT.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-36056B87im2IVkWE6.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-115142STcufDxOph3w.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-50474fw60BHa567Pj.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-72208thOrjHNwMYV8.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-61883cttD5QXdax8B.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-395379MOjLvE4RN6L0.json
/home/laura/.local/share/jupyter/runtime/kernel-v2-61883RisQBlScLSyj.json
find: ‘/home/laura/git/spring-ai-rag-expert/volumes/etcd/member’: Keine Berechtigung
find: ‘standard output’: Datenübergabe unterbrochen (broken pipe)
find: Schreibfehler.
laura@laura-ms7d42:~/git/spring-ai-intro$ ▯
```

More examples with “-s”

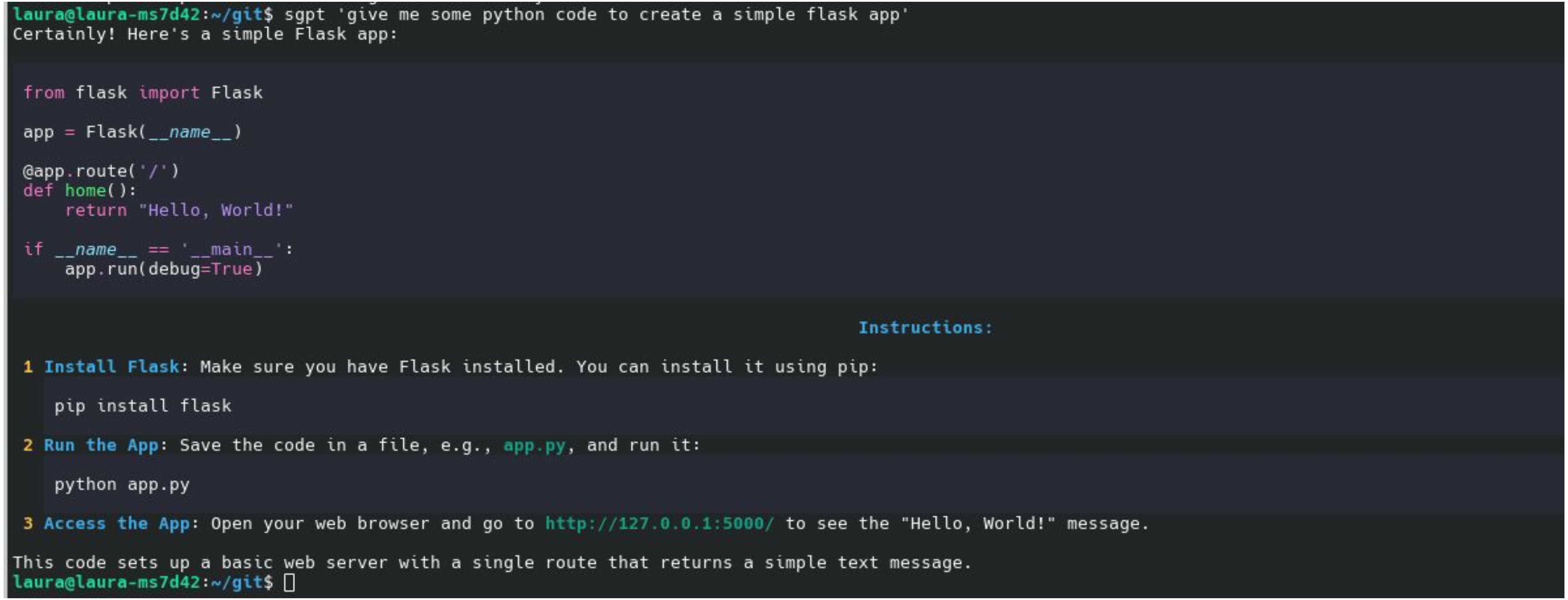
```
laura@laura-ms7d42:~/git/spring-ai-intro$ sgpt -s "update my system"
sudo apt update && sudo apt upgrade -y
[E]xecute, [D]escribe, [A]bort: a

laura@laura-ms7d42:~/git/spring-ai-intro$ sgpt -s "start nginx container, mount ./index.html"
docker run -d -v $(pwd)/index.html:/usr/share/nginx/html/index.html:ro -p 80:80 nginx
[E]xecute, [D]escribe, [A]bort:
```

Example 2 - Shell GPT for Kubernetes



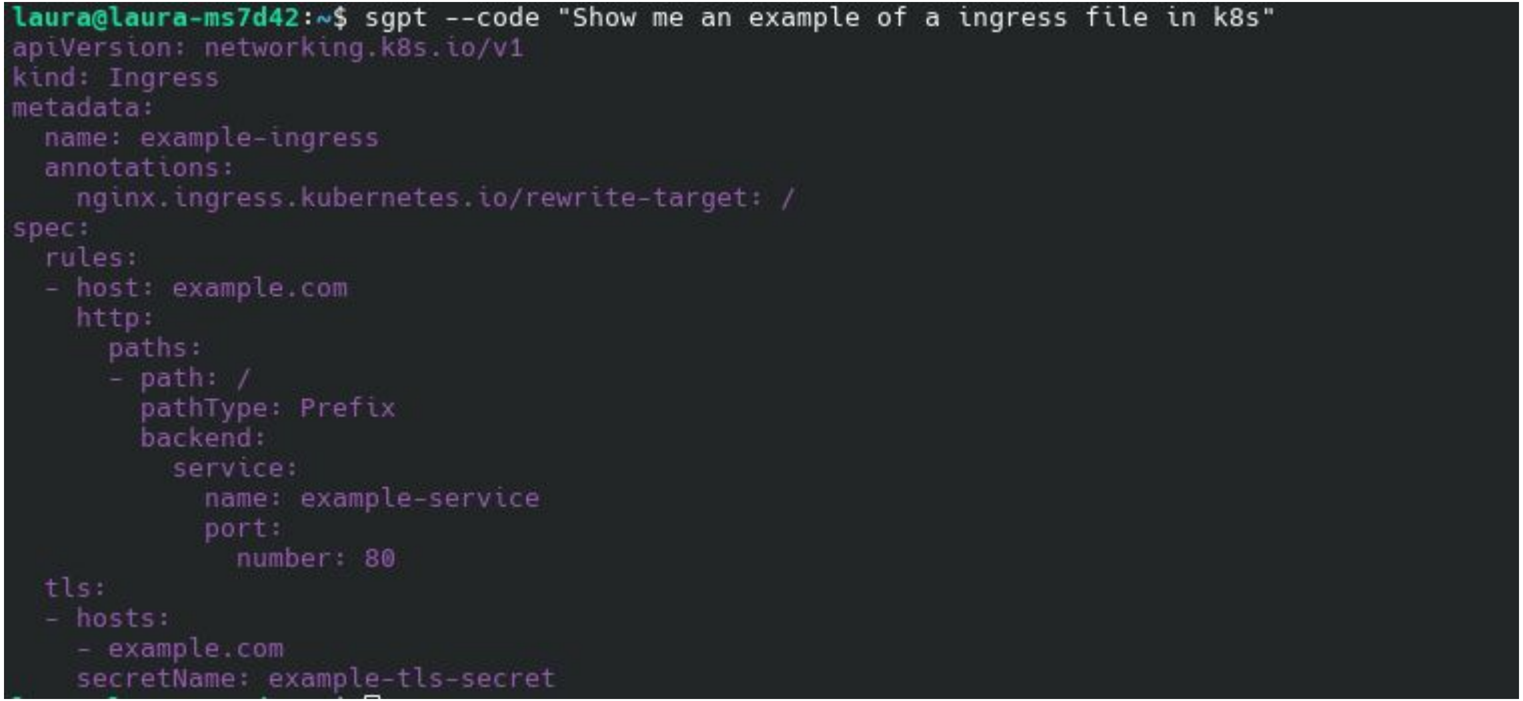
```
laura@laura-ms7d42:~/git$ sgpt 'give a kubernetes command for to create a stateful set with mysql'
To create a StatefulSet with MySQL in Kubernetes, you can use the following YAML configuration. Save it to a file, e.g., mysql-statefulset.yaml, and then apply it using
kubectl.

 apiVersion: apps/v1
 kind: StatefulSet
 metadata:
   name: mysql
 spec:
   serviceName: "mysql"
   replicas: 1
   selector:
     matchLabels:
       app: mysql
   template:
     metadata:
       labels:
         app: mysql
     spec:
       containers:
       - name: mysql
         image: mysql:5.7
         ports:
         - containerPort: 3306
           name: mysql
         env:
         - name: MYSQL_ROOT_PASSWORD
           value: yourpassword
   volumeClaimTemplates:
   - metadata:
       name: mysql-persistent-storage
     spec:
       accessModes: [ "ReadWriteOnce" ]
       resources:
         requests:
           storage: 1Gi

Apply the configuration with:

 kubectl apply -f mysql-statefulset.yaml

Make sure to replace yourpassword with a secure password.
```

Example 3 - Code generation



```
laura@laura-ms7d42:~/git$ sgpt 'give me some python code to create a simple flask app'
Certainly! Here's a simple Flask app:


 from flask import Flask

 app = Flask(__name__)

 @app.route('/')
 def home():
     return "Hello, World!"

 if __name__ == '__main__':
     app.run(debug=True)


                                                                                  Instructions:

 1 Install Flask: Make sure you have Flask installed. You can install it using pip:

    pip install flask

 2 Run the App: Save the code in a file, e.g., app.py, and run it:

    python app.py

 3 Access the App: Open your web browser and go to http://127.0.0.1:5000/ to see the "Hello, World!" message.

This code sets up a basic web server with a single route that returns a simple text message.
laura@laura-ms7d42:~/git$ 
```

The “--code” option



```
laura@laura-ms7d42:~$ sgpt --code "Show me an example of a ingress file in k8s"
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: example-service
            port:
              number: 80
  tls:
  - hosts:
    - example.com
    secretName: example-tls-secret
```

Chat mode (session)

To start a conversation, use the **`--chat`** option followed by a unique session name and a prompt.

```
> sgpt --chat conversation_1 "please remember my favorite number: 4"

# -> I will remember that your favorite number is 4.

> sgpt --chat conversation_1 "what would be my favorite number + 4?"

# -> Your favorite number is 4, so if we add 4 to it, the result would be 8.
```

You can use chat sessions to iteratively improve GPT suggestions by providing additional details. It is possible to use `--code` or `--shell` options to initiate `--chat`:

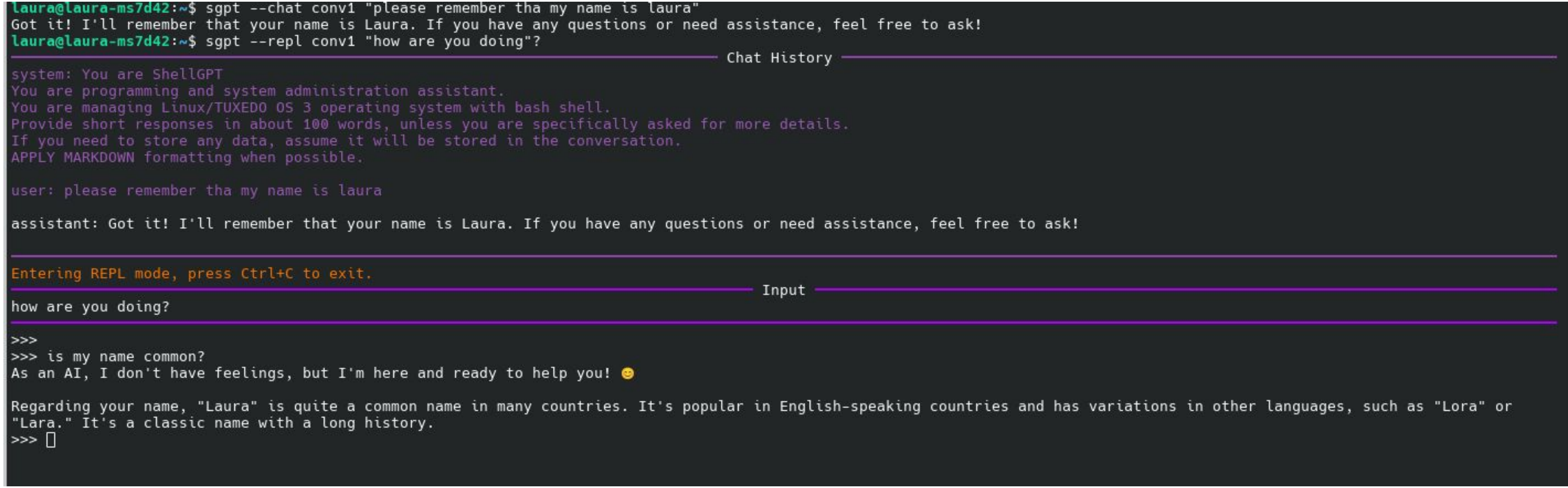
```
sgpt --chat conversation_2 --code "make a request to localhost using python"
import requests
```

REPL Mode

REPL (read–eval–print loop) mode allows you to interactively chat with GPT models. To start a chat session in REPL mode, use the `--repl` option followed by a unique session name. You can also use "temp" as a session name to start a temporary REPL session. Note that `--chat` and `--repl` are using same underlying object, so you can use `--chat` to start a chat session and then pick it up with `--repl` to continue the conversation in REPL mode.

```
sgpt --repl temp

Entering REPL mode, press Ctrl+C to exit.

>>> What is REPL?

REPL stands for Read-Eval-Print Loop. It is a programming environment ...

>>> How can I use Python with REPL?

To use Python with REPL, you can simply open a terminal or command prompt ...
```

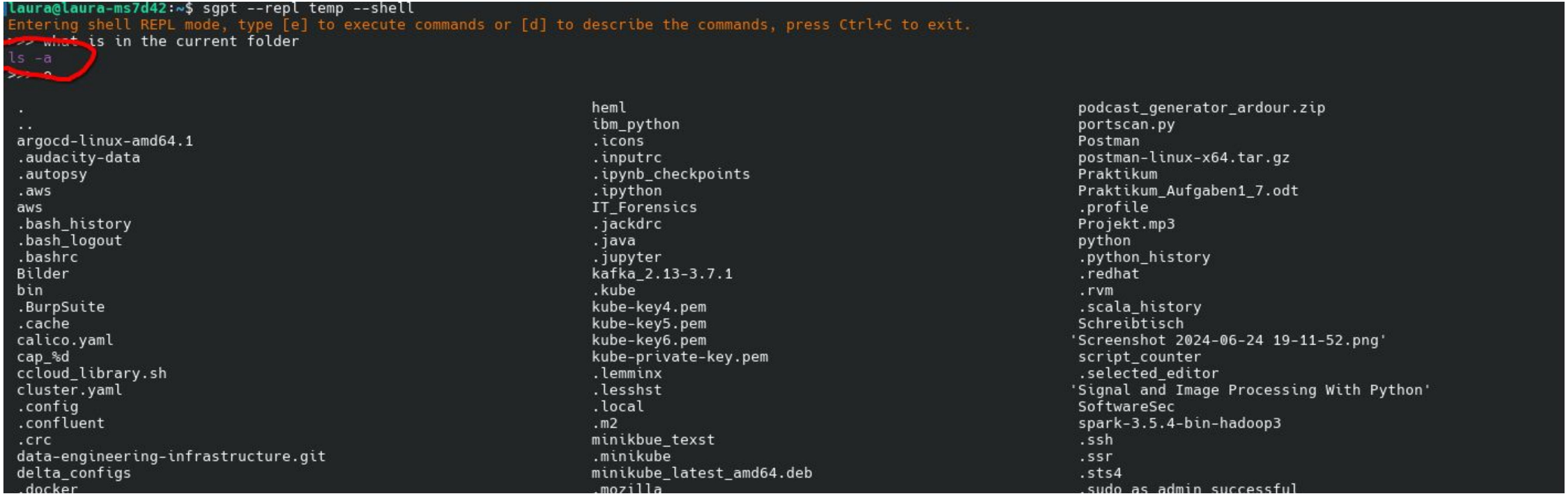
Example chat / repl mode



```
laura@laura-ms7d42:~$ sgpt --chat conv1 "please remember tha my name is laura"
Got it! I'll remember that your name is Laura. If you have any questions or need assistance, feel free to ask!
laura@laura-ms7d42:~$ sgpt --repl conv1 "how are you doing"?
─────────────────────────────────────────────── Chat History ───────────────────────────────────────────────
system: You are ShellGPT
You are programming and system administration assistant.
You are managing Linux/TUXEDO OS 3 operating system with bash shell.
Provide short responses in about 100 words, unless you are specifically asked for more details.
If you need to store any data, assume it will be stored in the conversation.
APPLY MARKDOWN formatting when possible.

user: please remember tha my name is laura

assistant: Got it! I'll remember that your name is Laura. If you have any questions or need assistance, feel free to ask!

────────────────────────────────────────────────────────────────────────────────────────────────────────────
Entering REPL mode, press Ctrl+C to exit.
────────────────────────────────────────────────── Input ───────────────────────────────────────────────────
how are you doing?
────────────────────────────────────────────────────────────────────────────────────────────────────────────
>>>
>>> is my name common?
As an AI, I don't have feelings, but I'm here and ready to help you! 😊

Regarding your name, "Laura" is quite a common name in many countries. It's popular in English-speaking countries and has variations in other languages, such as "Lora" or
"Lara." It's a classic name with a long history.
>>> ▯
```

Repl mode 2

REPL mode can work with `--shell` and `--code` options, which makes it very handy for interactive shell commands and code generation:
sgpt --repl temp --shell



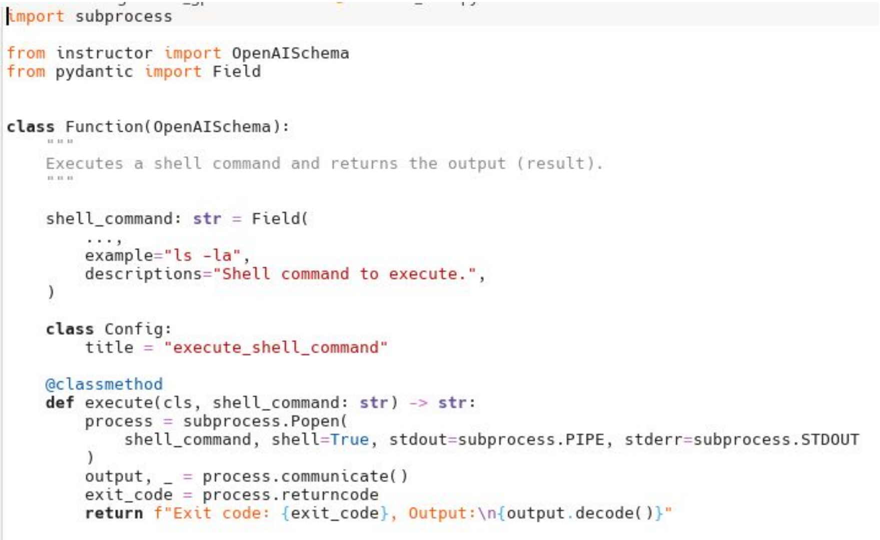
```
laura@laura-ms7d42:~$ sgpt --repl temp --shell
Entering shell REPL mode, type [e] to execute commands or [d] to describe the commands, press Ctrl+C to exit.
>>> what is in the current folder
ls -a
>>> e

 .                                                                heml                                                   podcast_generator_ardour.zip
 ..                                                               ibm_python                                             portscan.py
 argocd-linux-amd64.1                                             .icons                                                 Postman
 .audacity-data                                                   .inputrc                                               postman-linux-x64.tar.gz
 .autopsy                                                         .ipynb_checkpoints                                     Praktikum
 .aws                                                             .ipython                                               Praktikum_Aufgaben1_7.odt
 aws                                                              IT_Forensics                                           .profile
 .bash_history                                                    .jackdrc                                               Projekt.mp3
 .bash_logout                                                     .java                                                  python
 .bashrc                                                          .jupyter                                               .python_history
 Bilder                                                           kafka_2.13-3.7.1                                       .redhat
 bin                                                              .kube                                                  .rvm
 .BurpSuite                                                       kube-key4.pem                                          .scala_history
 .cache                                                           kube-key5.pem                                          Schreibtisch
 calico.yaml                                                      kube-key6.pem                                         'Screenshot 2024-06-24 19-11-52.png'
 cap_%d                                                           kube-private-key.pem                                   script_counter
 ccloud_library.sh                                                .lemminx                                               .selected_editor
 cluster.yaml                                                     .lesshst                                              'Signal and Image Processing With Python'
 .config                                                          .local                                                 SoftwareSec
 .confluent                                                       .m2                                                    spark-3.5.4-bin-hadoop3
 .crc                                                             minikbue_texst                                         .ssh
 data-engineering-infrastructure.git                              .minikube                                              .ssr
 delta_configs                                                    minikube_latest_amd64.deb                              .sts4
 .docker                                                          .mozilla                                               .sudo_as_admin_successful
```

Function calling

Install default functions.

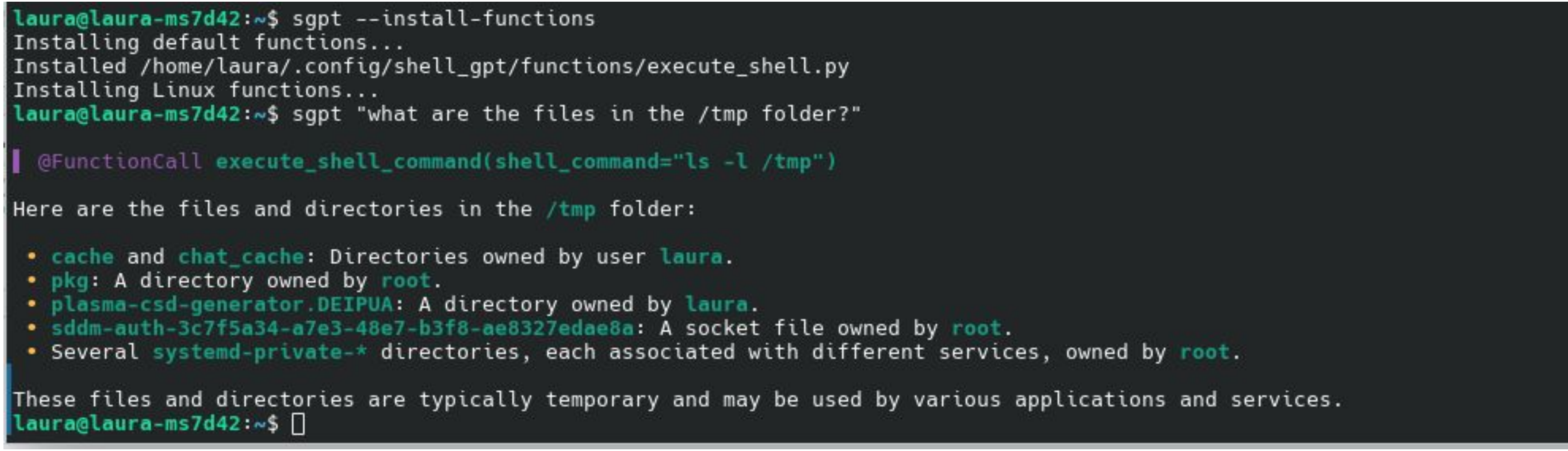
```
> sgpt --install-functions
```

Add custom functions.
navigate to `~/.config/shell_gpt/functions` and create a new .py file with the function name. Inside this file, you can define your function using the following syntax



```python
import subprocess

from instructor import OpenAISchema
from pydantic import Field


class Function(OpenAISchema):
    """
    Executes a shell command and returns the output (result).
    """

    shell_command: str = Field(
        ...,
        example="ls -la",
        descriptions="Shell command to execute.",
    )

    class Config:
        title = "execute_shell_command"

    @classmethod
    def execute(cls, shell_command: str) -> str:
        process = subprocess.Popen(
            shell_command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT
        )
        output, _ = process.communicate()
        exit_code = process.returncode
        return f"Exit code: {exit_code}, Output:\n{output.decode()}"
```

Example with function call



```
laura@laura-ms7d42:~$ sgpt --install-functions
Installing default functions...
Installed /home/laura/.config/shell_gpt/functions/execute_shell.py
Installing Linux functions...
laura@laura-ms7d42:~$ sgpt "what are the files in the /tmp folder?"

▌ @FunctionCall execute_shell_command(shell_command="ls -l /tmp")

Here are the files and directories in the /tmp folder:

 • cache and chat_cache: Directories owned by user laura.
 • pkg: A directory owned by root.
 • plasma-csd-generator.DEIPUA: A directory owned by laura.
 • sddm-auth-3c7f5a34-a7e3-48e7-b3f8-ae8327edae8a: A socket file owned by root.
 • Several systemd-private-* directories, each associated with different services, owned by root.

These files and directories are typically temporary and may be used by various applications and services.
laura@laura-ms7d42:~$ 
```

Roles

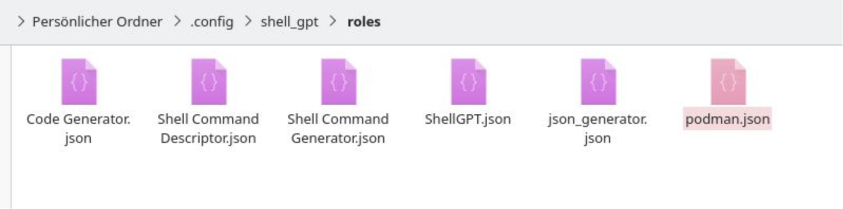
ShellGPT allows you to create custom roles, which can be utilized to generate code, shell commands, or to fulfill your specific needs.

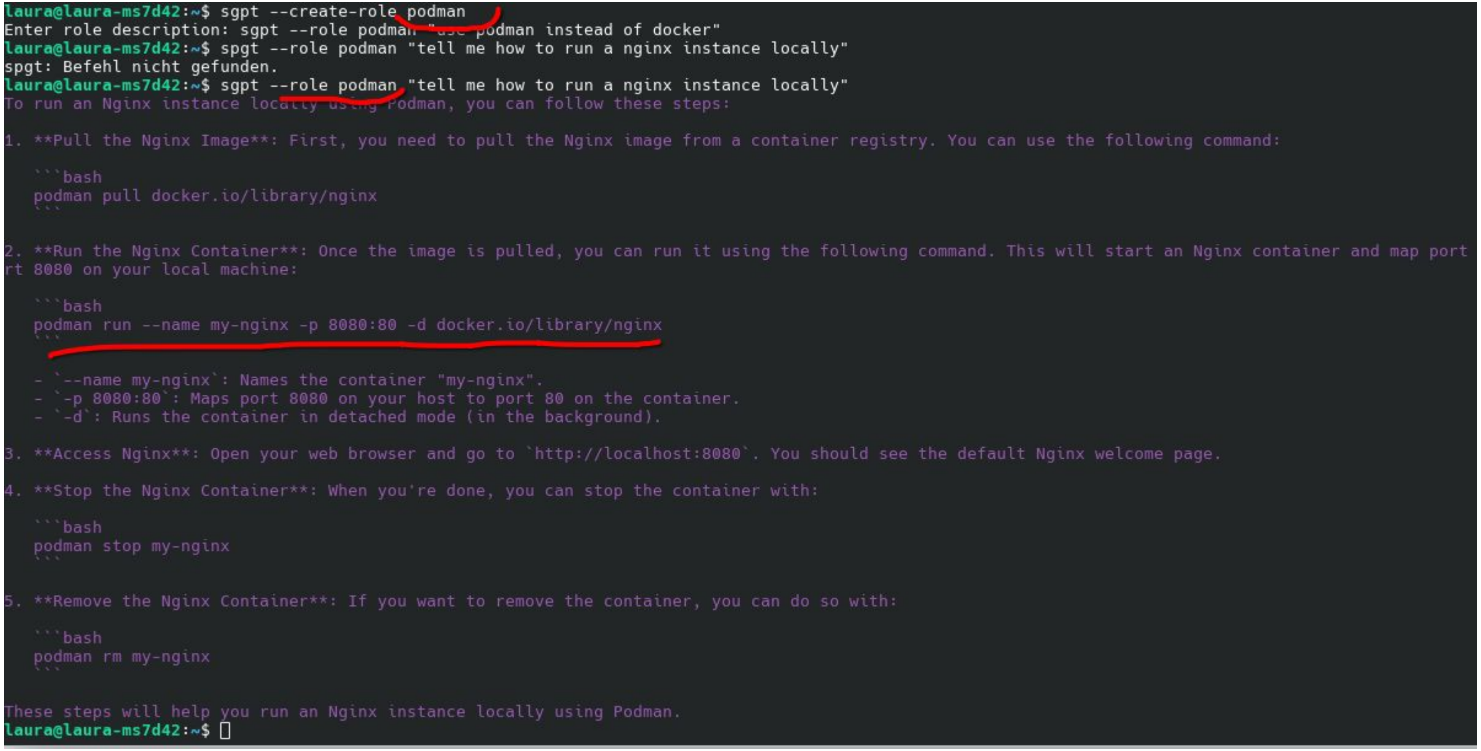
To create a new role, use the `--create-role` option followed by the role name.

You will be prompted to provide a description for the role, along with other details. This will create a JSON file in `~/.config/shell_gpt/roles` with the role name.

Inside this directory, you can also edit default `sgpt` roles, such as shell, code, and default. Use the `--list-roles` option to list all available roles, and the `--show-role` option to display the details of a specific role.

Role - Example





~~~
laura@laura-ms7d42:~$ sgpt --create-role podman
Enter role description: sgpt --role podman "use podman instead of docker"
laura@laura-ms7d42:~$ spgt --role podman "tell me how to run a nginx instance locally"
spgt: Befehl nicht gefunden.
laura@laura-ms7d42:~$ sgpt --role podman "tell me how to run a nginx instance locally"
To run an Nginx instance locally using Podman, you can follow these steps:

1. **Pull the Nginx Image**: First, you need to pull the Nginx image from a container registry. You can use the following command:

   ```bash
   podman pull docker.io/library/nginx
   ~~~

2. **Run the Nginx Container**: Once the image is pulled, you can run it using the following command. This will start an Nginx container and map port
rt 8080 on your local machine:

   ```bash
   podman run --name my-nginx -p 8080:80 -d docker.io/library/nginx
   ```

   - `--name my-nginx`: Names the container "my-nginx".
   - `-p 8080:80`: Maps port 8080 on your host to port 80 on the container.
   - `-d`: Runs the container in detached mode (in the background).

3. **Access Nginx**: Open your web browser and go to `http://localhost:8080`. You should see the default Nginx welcome page.

4. **Stop the Nginx Container**: When you're done, you can stop the container with:

   ```bash
   podman stop my-nginx
   ```

5. **Remove the Nginx Container**: If you want to remove the container, you can do so with:

   ```bash
   podman rm my-nginx
   ```

These steps will help you run an Nginx instance locally using Podman.
laura@laura-ms7d42:~$
```
```

Webpages about Shell GPT

https://www.toolify.ai/ai-news/install-shellgpt-in-ubuntu-linux-terminal-easy-guide-1920997

https://medium.com/@tech_18484/how-to-run-chatgpt-using-shellgpt-from-the-terminal-51e686a9a111

https://www.webasha.com/blog/what-is-shellgpt-why-every-hackers-need-to-use-shell-gpt-in-kali-linux

Github repositories

Python / Bash

- https://github.com/TheR1D/shell_gpt

Powershell

- https://github.com/yamautomate/ShellGPT

Thank you!