Enhanced LLM completions in the Zed editor with the Model Context Protocol (MCP)

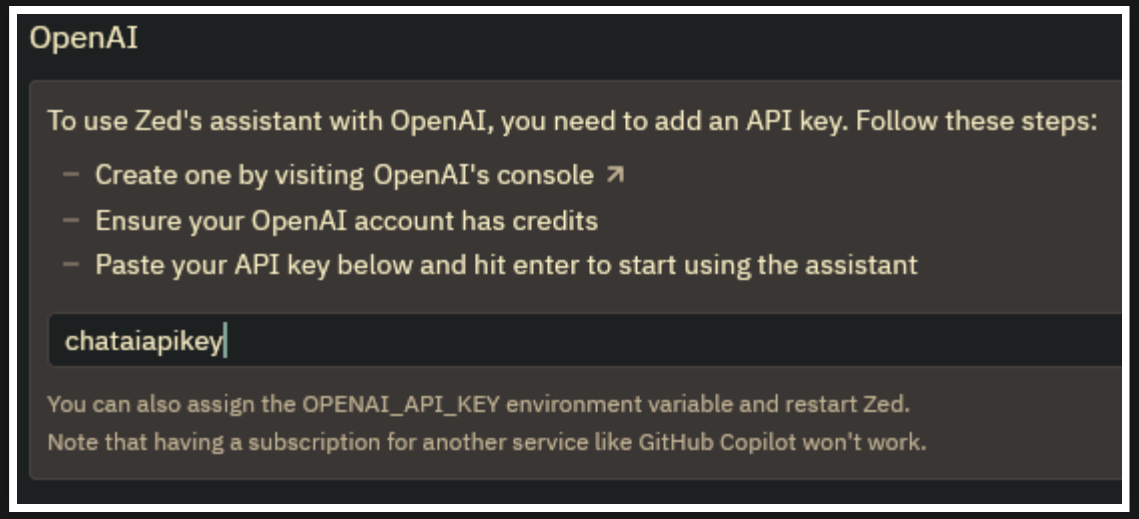
Zed

- https://zed.dev
- Fast VSCode successor
- builtin support for LLMs and MCP

LLM Completions in Zed with Chat-AI

Configuration

```
{ "language_models": {"openai": {
    "api_url": "https://chat-ai.academiccloud.de/v1",
    "available_models": [
      { "name": "meta-llama-3.1-8b-instruct",
        "max_tokens": 128000 },
      { "name": "qwen2.5-coder-32b-instruct",
        "max_tokens": 128000 }],
    "version": "1" },
  "assistant": { "version": "2",
    "default_model": {
      "provider": "openai",
      "model": "qwen2.5-coder-32b-instruct"}}}}
```


OpenAI
To use Zed's assistant with OpenAI, you need to add an API key. Follow these steps:
– Create one by visiting OpenAI's console ↗
– Ensure your OpenAI account has credits
– Paste your API key below and hit enter to start using the assistant
chataiapikey
You can also assign the OPENAI_API_KEY environment variable and restart Zed.
Note that having a subscription for another service like GitHub Copilot won't work.

Basics

Assistant panel is Chat interface

- Used as context for completions
- Editable
- Slash Commands
  - `/terminal` Terminal output
  - `/diagnostics` insert diagnostics
  - `/fetch` inserts webpage as markdown

`Ctrl+Enter` for Inline Assistant

- Editors
- Terminal

Demos

- fix config
- generate terminal command
- implement function
    - write test

Model Context Protocol (MCP)

- https://modelcontextprotocol.io/
- protocol to provide tools and ressources to LLMs
- developed by Anthropic
- similar to Language Server Protocol (LSP)
- uses JSON-RPC 2.0
- Client/Server
- transport: bidirectional byte-stream / stdio
    - or HTTP

Types of Requests

- Ressources
  - Get data for LLM Context
- Prompts
- Tools
  - Side-Effects / Compute / Do

Applications

- awesome-mcp-servers
- “official” servers
  - memory
  - git
  - fetch
  - filesystem
- docker
- web search
- code execution - **We are working on this!**

MCP in Zed

- https://zed.dev/blog/mcp
- Add Slash Commands for MCP server functions
- only prompts supported for now

```
{"context_servers":{
  "mcp-server": {
    "command": {
      "path": "uv",
      "args": ["--directory",
        "/path/to/mcp-server",
        "run", "mcp", "run", "server.py"]
    },
    "settings": {}
  }
}}
```

A simple MCP server:

```python
# inspired by
# https://github.com/modelcontextprotocol/python-sdk
from mcp.server.fastmcp import FastMCP
import uuid as uid
mcp = FastMCP("Demo")
@mcp.prompt()
def uuid() -> str:
    return f"a uuidv4 for you: {uid.uuid4()}"
```

MCP Inspector from python-sdk useful for debugging