



# Qdrant MCP

## Semantic Code Lookup for AI Assistants

Evgeniya Sukhodolskaya,  
Developer Advocate,  
Qdrant

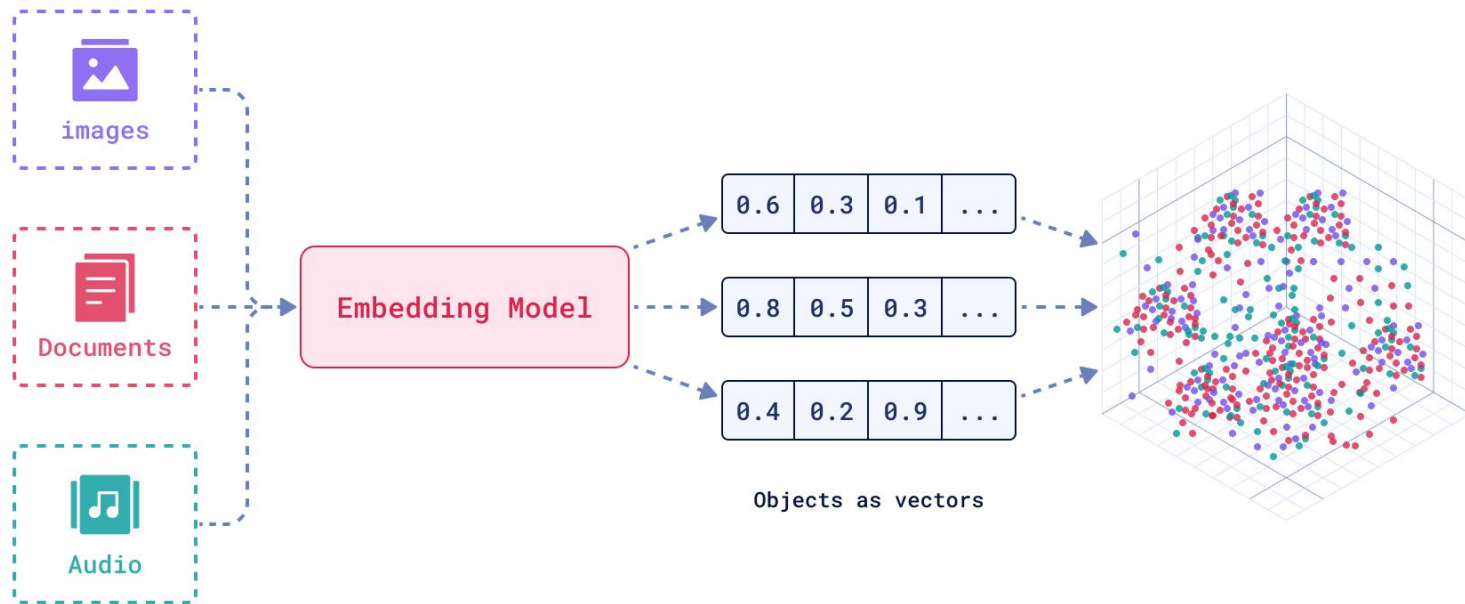


# Qdrant & Semantic Lookup

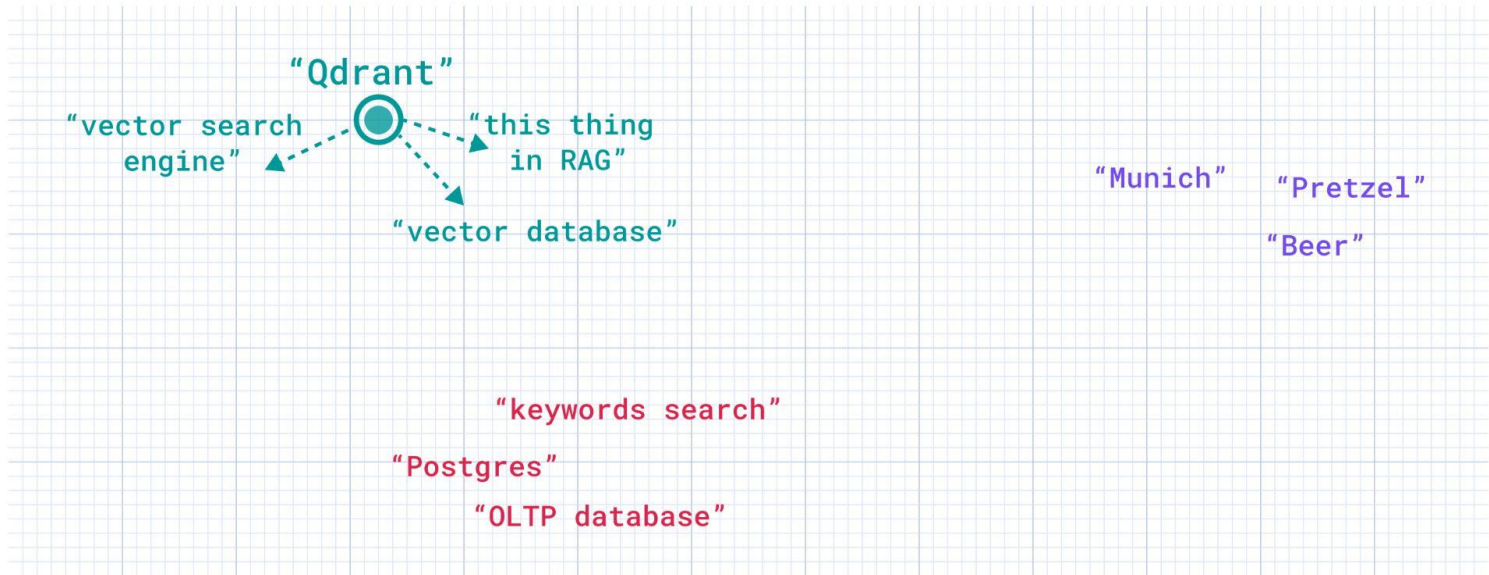
# Qdrant is an open-source Vector Search Engine

or you might have also heard  
the term “*vector database*”,  
“*semantic similarity search engine*”,  
“*this thing in (Agentic) RAG*”

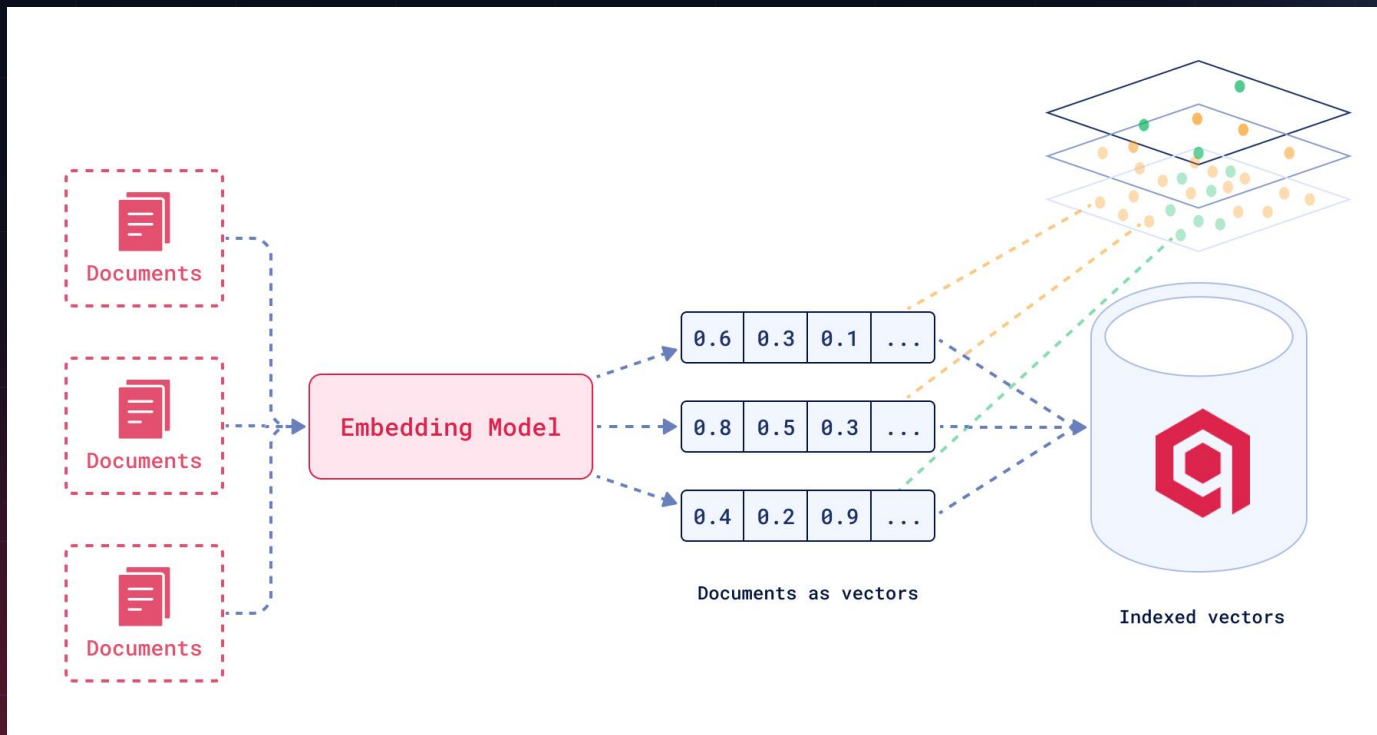
# Qdrant is a **Vector** Search Engine



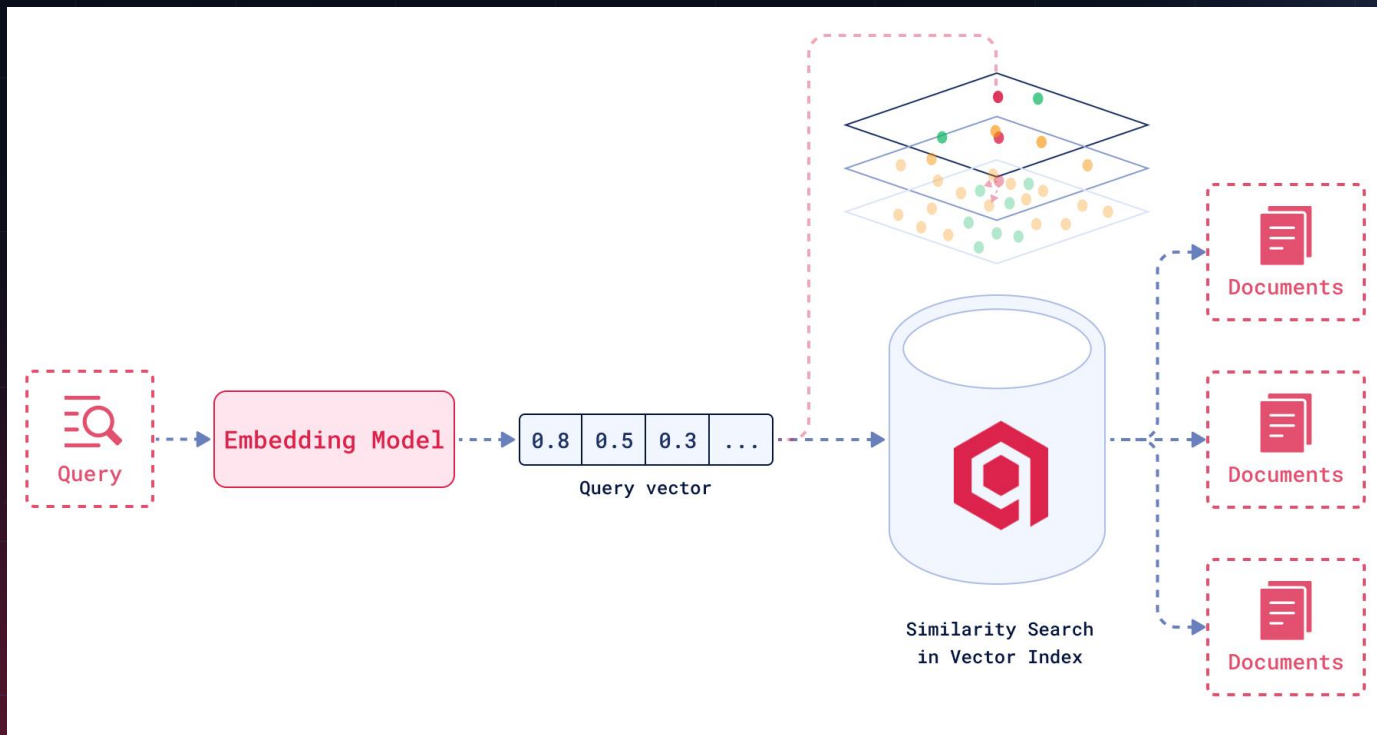
# Qdrant is a **Vector Search** Engine



# Qdrant is a **Vector Search Engine**: Store



# Qdrant is a **Vector Search Engine**: Find



# Sources for Ones Interested



Article “*An Introduction to Vector Databases*” –  
an overview of all relevant concepts;

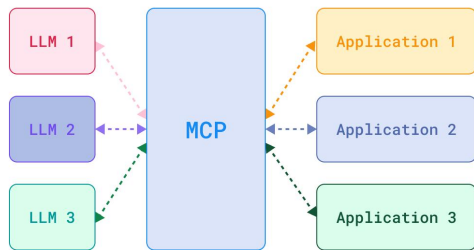
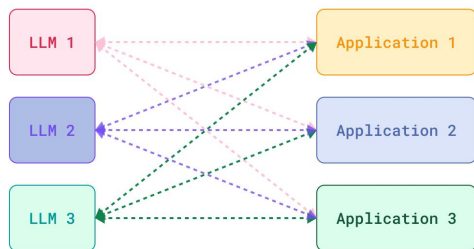


Article “*Built for Vector Search*” –  
why vector search needs a dedicated solution.



# Qdrant MCP Server

# Model Context Protocol (MCP)



*“MCP is an open protocol that standardizes how applications provide context to LLMs.*

*Think of MCP like a USB-C port for AI applications.”*

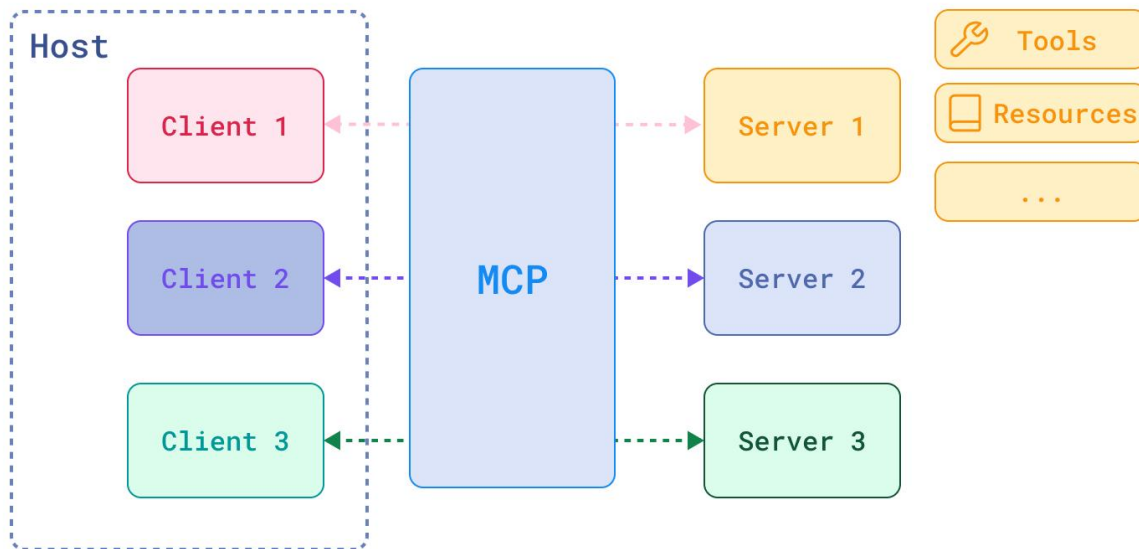


[modelcontextprotocol.io](https://modelcontextprotocol.io)

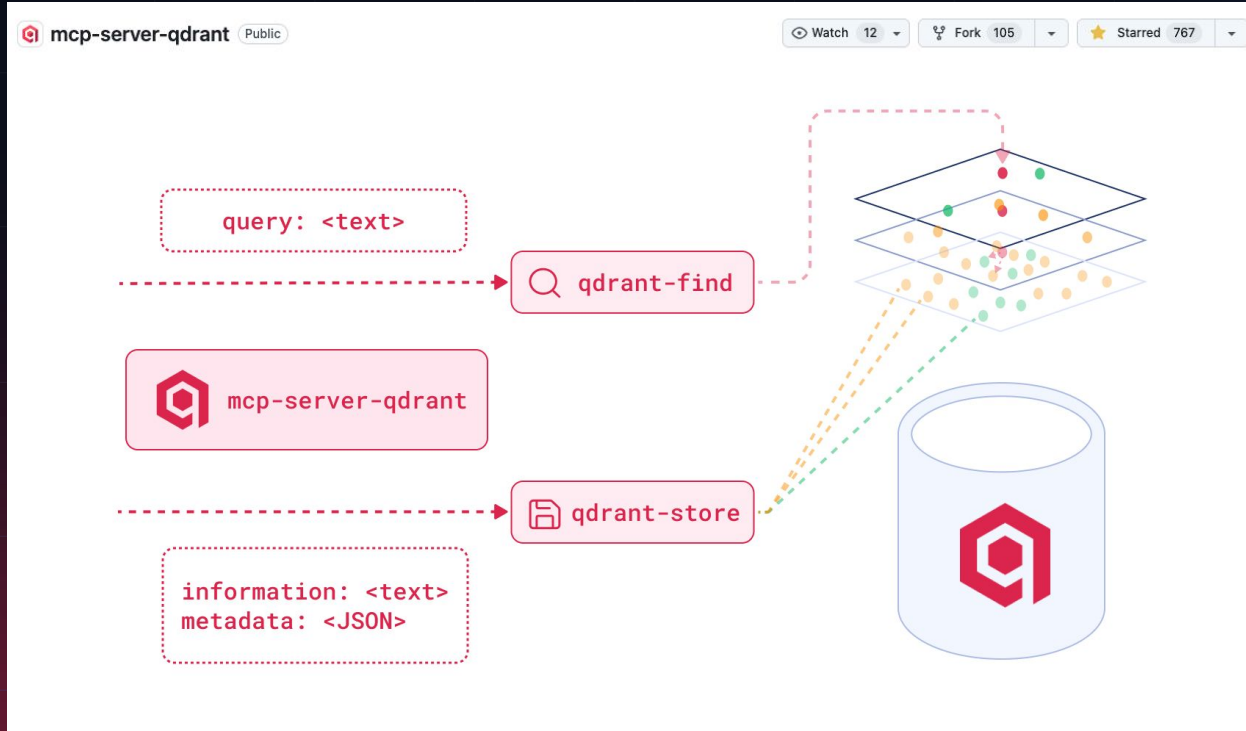


[HuggingFace MCP course](#)

# Model Context Protocol (MCP)



# Qdrant MCP Server



mcp-server-qdrant



# Possible Applications

- Inline Retrieval-Augmented Generation (RAG) <originally used as a note-taker>;
- Automating your codebase documentation;
- Personalizing your code assistant with best practices, templates, and specifics of your codebase;
- (today's talk) Providing API reference and code documentation for agents.


# How it all Started

## User:

- *Implement hybrid search in Qdrant*

## Coding Assistant:

- Uses deprecated method “search” instead of “query\_points”;
- Doesn't know Qdrant, since 1.10.0, supports combining searches with “prefetch” & fusion;
- Doesn't know Qdrant can calculate Inverse Document Frequency (IDF) on the server side;
- Doesn't know about local & cloud inference...

=> People come & say: “Qdrant doesn't have hybrid search functionality” 

# How it all Started

So, AI coding assistant fails to implement hybrid search with Qdrant (and it's even one of the widely known applications).

## Why?

LLMs are retrained, but it's mostly **impossible to keep them** constantly **updated & in sync** with specific library versions.

## And Qdrant has:

- **Several clients** (Java, Python, GO, C#, Rust, Typescript);
- **Several versions** (1.15 comes out this week, btw), each with new features.

## LLM-friendly API reference?

Yes, we have **/llm-full.txt** at our API Reference page, but it's big (~**53K tokens**), so models will still have troubles to generate desired code when fed such a large context.

**Idea:**  
**mcp-server-qdrant**  
**as API Reference for**  
**Agents**



# What do We Want

We need, for all package functions, to **store**:

- Well-curated snippet;
- Accurate description;
- Programming language tag;
- Version tag;
- Package name tag (*who said we'll stop at Qdrant if it works?*).

We need, for every user request, to **find**:

- **Semantic match** by description of a code snippet,  
**filtered** by fitting version, package name, and programming language

# At the Same Time AI Assistant

**qdrant-store:**  
sometimes stores  
weird metadata

**qdrant-find:**  
troubles with  
consistent &  
correctly-formed  
filters

Point 420e1264-06c0-4b28-b41a-30b61a626668

Payload:

document

```
"""python api_key = os.getenv("QDRANT_API_KEY") cloud_url = os.getenv("QDRANT_URL") return QdrantClient( url=cloud_url, api_key=api_key, ) """  
Basic Qdrant client initialization using environment variables. Retrieves API key and cloud URL from environment variables and creates a QdrantClient  
instance for connecting to Qdrant Cloud.
```

metadata

```
{  
  "code_type": "client_initialization"  
  "authentication": "api_key"  
  "environment_variables": [  
    0: "QDRANT_API_KEY"  
    1: "QDRANT_URL"  
  ]  
  "platform": "qdrant_cloud"  
}
```

???

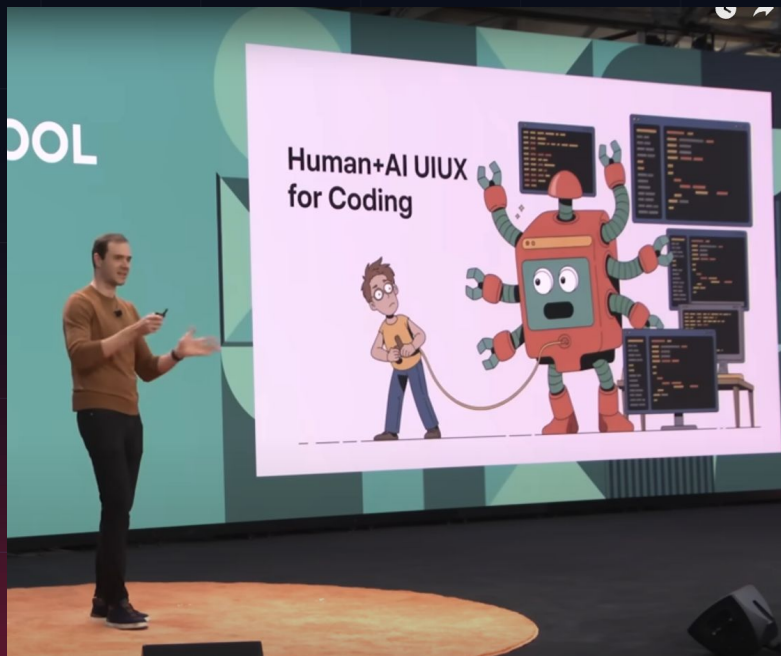
Vectors:

Vectors:

Name: fast-bge-base-en-v1.5 Length: 768

OPEN GRAPH FIND SIMILAR

# Keep Agents on a Leash



We need to help our AI coding assistants by avoiding situations with **too many degrees of freedom**.

So, we updated our **mcp-server-qdrant** to use it as a **customizable base class**:

- **Customizable filters** restrictions & LLM-friendly format of metadata filtering
- Customizable switching `qdrant-store` tool on/off (**`read_only` mode**)

And built on top our **own custom MCP server, suitable solely as an API reference for AI coding assistants.**

# #1 Remote MCP server: mcp-for-docs

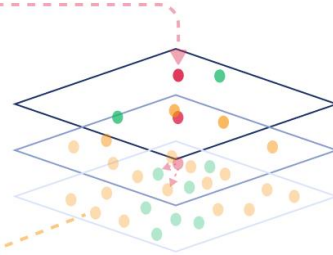
```
query: <text>
filter: {
  language==python,
  package==qdrant,
  version==1.15.0
}
```

🔍 qdrant-find



mcp-for-docs

```
metadata: {
  description: <text>
  language: python
  package name: qdrant
  package version: 1.15.0
  code snippet: <code>
}
```

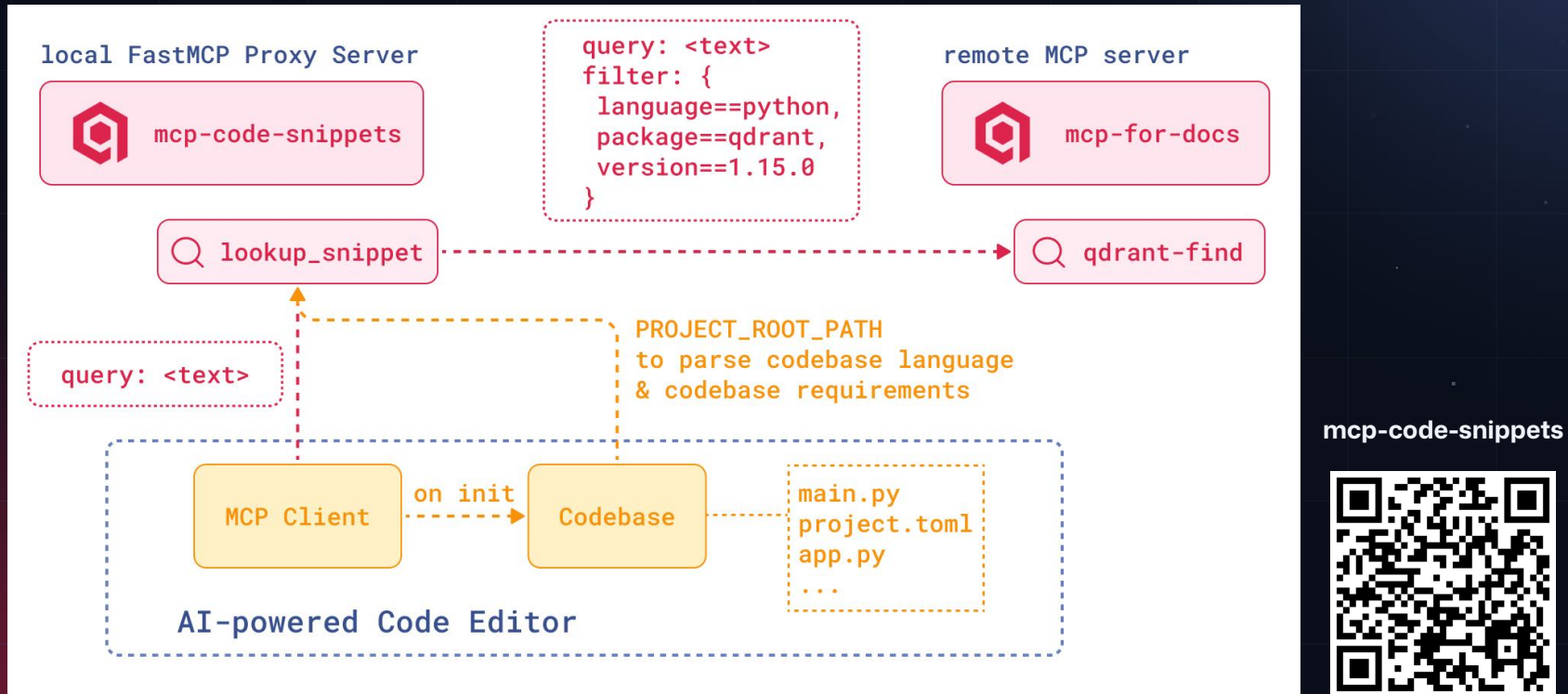


curated  
code  
snippets

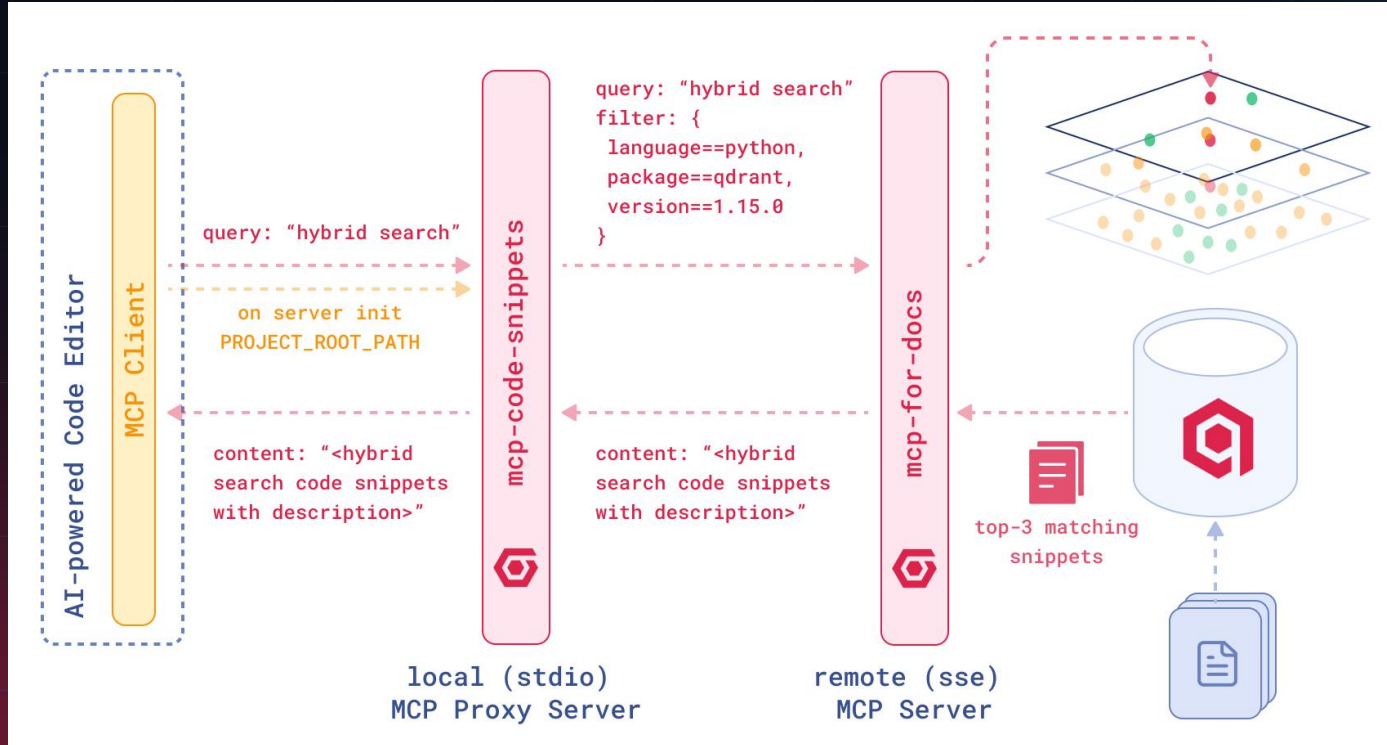
mcp-for-docs



# #2 Local MCP Proxy: mcp-for-docs



# How Does it Work



# How to Install

{ } mcp.json U •

.vscode > { } mcp.json > ...

```
1  {
2  |   "servers": {
3  |       ✓Running | Stop | Restart | Debug | 1 tools | More...
4  |       "mcpCodeSnippets": {
5  |           "command": "uvx",
6  |           "args": ["mcp-code-snippets"],
7  |           "env": {
8  |               "MCP_PROXY_CONFIG": "/Users/evgeniya_sukhodolskaya/Desktop/Qdrant/mcp-test/.vscode/mcp_proxy.json",
9  |               "PROJECT_ROOT_PATH": "/Users/evgeniya_sukhodolskaya/Desktop/Qdrant/mcp-test"
10 |           }
11 |       }
12 |   }
```

{ } mcp.json U • { } mcp\_proxy.json U •

.vscode > { } mcp\_proxy.json > ...

```
1  {
2  |   "mcpServers": {
3  |       "mcp-server-qdrant": {
4  |           "url": "https://mcp.qdrant.tech/mcp/",
5  |           "transport": "streamable-http"
6  |       }
7  |   }
8  }
```

# What's Now & What's Next

## What's Now:

`mcp-code-snippets` can be installed and works with Qdrant code snippets of the latest version.

*So if you use it, hybrid search case will work—>*

## Plans:

- Further development, publishing & maintenance;
- Package versioning support;
- Support for other packages;
- Further development of the base `mcp-server-qdrant` server.

## Goal:

Turn `mcp-code-snippets` into a **universal open-source API reference for AI coding assistants**.

I'll help you create a simple example of hybrid search in Qdrant. Let me first look up code snippets for Qdrant hybrid search functionality.

```
> Ran lookup-snippet – mcpCodeSnippets ✓  
(MCP Server)
```

Great! Now let me look up more specific information about setting up collections for hybrid search:

```
> Ran lookup-snippet – mcpCodeSnippets ✓  
(MCP Server)
```

Now let me look up information about adding points with both dense and sparse vectors:

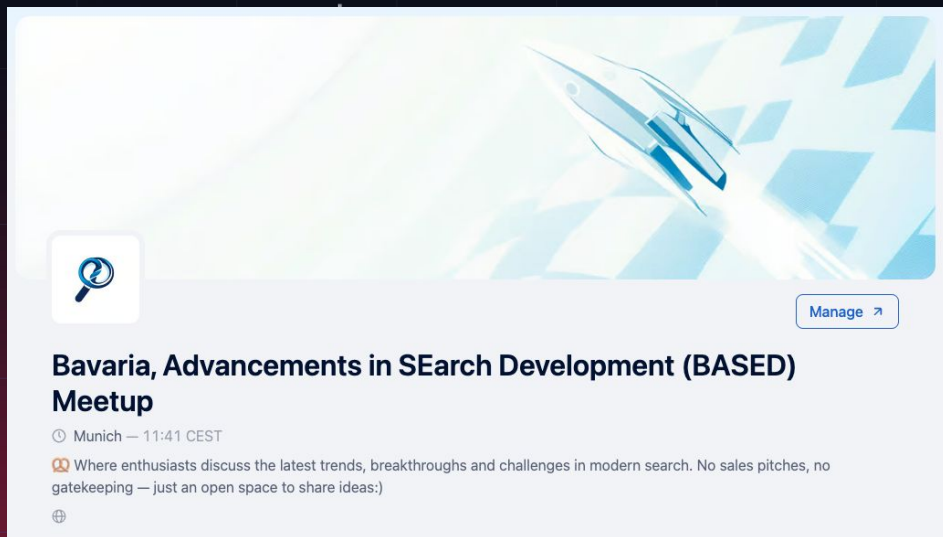
```
> Ran lookup-snippet – mcpCodeSnippets ✓  
(MCP Server)
```



# Thank You!

## Let's Keep the Tech Yapping Going:)

### My Munich Search Meetup



[lu.ma/based\\_meetup](https://lu.ma/based_meetup)



My LinkedIn ->  
Evgeniya  
Sukhodolskaya

