Multi-Agent Software Engineering on RTX 4070 8GB: Complete Implementation Guide

PART 1: OPEN SOURCE MODEL RECOMMENDATIONS

🥇 TOP CODE GENERATION MODELS

Qwen2.5-Coder-7B-Instruct ⭐ BEST OVERALL

- **Parameters:** 7.61B (6.53B non-embedding)
- **Context Length:** 128K tokens
- **Release Date:** September 2024
- **License:** Apache 2.0

Quantized VRAM Requirements:

- Q4_0 (4-bit): **3.8 GB** ✅
- Q5_0 (5-bit): **4.75 GB** ✅
- Q8_0 (8-bit): **7.61 GB** ✅

Benchmarks:

- **HumanEval:** 84.1% (Instruct)
- **HumanEval (Base):** 61.6%
- **MBPP:** Strong scores
- **BigCodeBench:** SOTA for size class

HuggingFace Links:

- Base: https://huggingface.co/Qwen/Qwen2.5-Coder-7B
- Instruct: https://huggingface.co/Qwen/Qwen2.5-Coder-7B-Instruct
- GGUF: https://huggingface.co/Qwen/Qwen2.5-Coder-7B-Instruct-GGUF

Why Best: Highest HumanEval scores (84.1%), Apache 2.0 license, 128K context, fits comfortably with Q4/Q5 quantization, trained on 5.5T tokens with FIM support. AI-SCHOLAR MarkTechPost

---

StarCoder2-7B 🥈 BEST ALTERNATIVE

- **Parameters:** 7B
- **Context Length:** 16,384 tokens (sliding window 4,096)

- **Release Date:** February 2024
- **License:** BigCode OpenRAIL-M v1

Quantized VRAM Requirements:

- Q4_K_M: **4.4 GB** ✅
- Q5_0: **4.94 GB** ✅
- Q8_0: **7.63 GB** ✅

Benchmarks:

- **HumanEval:** 35.4%
- **HumanEval+:** 29.9%
- **RepoBench:** 72.07%
- **GSM8K:** 40.4%

HuggingFace Links:

- Base: https://huggingface.co/bigcode/starcoder2-7b
- GGUF: https://huggingface.co/second-state/StarCoder2-7B-GGUF

Why Alternative: Open data provenance, permissive license, excellent repository-level understanding, 17 programming languages, trained on 3.5T tokens. Hugging Face +2

Qwen2.5-Coder-3B-Instruct 🥉 BEST LIGHTWEIGHT

- **Parameters:** 3.09B (2.77B non-embedding)
- **Context Length:** 32K tokens
- **Release Date:** November 2024
- **License:** Qwen Research License (restricted)

Quantized VRAM Requirements:

- Q4_0: **1.55 GB** ✅
- Q5_0: **1.93 GB** ✅
- Q8_0: **3.09 GB** ✅

Benchmarks:

- **MBPP:** Higher than 1.5B variant
- **LiveCodeBench:** Competitive

HuggingFace Links:

- Base: https://huggingface.co/Qwen/Qwen2.5-Coder-3B
- Instruct: https://huggingface.co/Qwen/Qwen2.5-Coder-3B-Instruct

Why Lightweight: Only 1.5-2GB with Q4, leaves 6GB+ for agents/context, excellent performance for size.

Additional Code Generation Models

DeepSeek-Coder-V2-Lite (16B MoE, 2.4B active)

- **Parameters:** 16B total, 2.4B active (MoE)
- **Context:** 128K tokens
- **License:** MIT
- **HumanEval:** 81.1% (Instruct)
- **MBPP+:** 68.8%
- **Link:** https://huggingface.co/deepseek-ai/DeepSeek-Coder-V2-Lite-Instruct
- **Note:** MoE architecture challenging for 8GB—requires careful memory management

CodeLlama-7B

- **Parameters:** 7B
- **Context:** 16K tokens
- **License:** Llama 2 Community License
- **HumanEval:** 50.6% (Instruct)
- **Q4_0:** 4.0 GB ✅
- **Links:**
  - Base: https://huggingface.co/codellama/CodeLlama-7b-hf
  - Instruct: https://huggingface.co/codellama/CodeLlama-7b-Instruct-hf
  - GGUF: https://huggingface.co/TheBloke/CodeLlama-7B-GGUF
- **Note:** 2023 model, outperformed by 2024 alternatives

StarCoder2-3B

- **Parameters:** 3B
- **HumanEval:** 31-33%
- **Q4_0:** 1.7 GB ✅
- **Link:** https://huggingface.co/bigcode/starcoder2-3b

- **GGUF:** https://huggingface.co/second-state/StarCoder2-3B-GGUF

🎯 TOP REASONING MODELS FOR COORDINATION

Llama 3.1 8B Instruct ⭐ BEST ALL-ROUNDER

- **Parameters:** 8B
- **Context Length:** 128K tokens
- **Release Date:** July 23, 2024
- **License:** Llama 3.1 Community License (permissive under 700M MAU) Hugging Face +2

Quantized VRAM Requirements:

- Q4_K_M: **4.9 GB** ✅
- Q5_K_M: **5.7 GB** ✅ (recommended)
- Q8_0: **6.6 GB** ✅

Benchmarks:

- **MMLU:** 69.4%
- **HumanEval:** 72.6%
- **MBPP++:** 72.8%
- **GSM8K:** 84.5%
- **IFEval:** 80.4% huggingface

HuggingFace Link: https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

Special Features: Tool calling, function support, 128K context, multilingual (8 languages), excellent for both coordination and execution. huggingface GitHub

---

Phi-3 Mini (3.8B) - 128K Context

- **Parameters:** 3.8B
- **Context:** 128K tokens
- **License:** MIT (fully permissive)
- **Release:** April 2024

Quantized VRAM:

- Q4_K_M: **2.5-3 GB** ✅

- Q8_0: **4 GB** ✅ (recommended)

Benchmarks:

- **MMLU:** 68.8%
- **HumanEval:** 59-61%
- **MBPP:** 70.0%
- **GSM8K:** 82.5% Continuumlabs LeMagIT

HuggingFace: https://huggingface.co/microsoft/Phi-3-mini-128k-instruct

Best For: Coordination tasks with long context, leaves 4-5GB for other components.

Gemma 2 9B Instruct

- **Parameters:** 9B
- **Context:** 8K tokens
- **License:** Gemma Terms of Use
- **Release:** June 2024 VentureBeat Google AI

Quantized VRAM:

- Q4_K_M: **5.5 GB** ✅
- Q5_K_M: **6.5 GB** ✅

HuggingFace: https://huggingface.co/google/gemma-2-9b-it

Features: Efficient architecture, sliding window attention, outperforms many 2x larger models. Google AI

Lightweight Reasoning Models

Llama 3.2 3B Instruct

- **Parameters:** 3B
- **Context:** 128K tokens
- **Q8_0:** 3.5 GB ✅
- **MMLU:** 63.4%
- **IFEval:** 77.4% (excellent instruction following) Medium
- **Link:** https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct Meta +2

Llama 3.2 1B Instruct

- **Parameters:** 1B
- **Context:** 128K tokens
- **Q8_0:** 1.8 GB ✅ (leaves 6GB+ free)
- **Link:** https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct
- **Best For:** Multiple agent instances, lightweight routing Meta AWS

Gemma 2 2B

- **Parameters:** 2.6B
- **Context:** 8K tokens
- **Q8_0:** 3 GB ✅
- **Link:** https://huggingface.co/google/gemma-2-2b-it Hugging Face Google Developers

---

Advanced Reasoning Models

Phi-3 Small (7B)

- **Parameters:** 7B
- **Context:** 128K tokens
- **License:** MIT
- **MMLU:** 75.3%
- **Q5_K_M:** 5.5 GB ✅
- **Link:** https://huggingface.co/microsoft/Phi-3-small-128k-instruct

Mistral NeMo 12B Instruct

- **Parameters:** 12B
- **Context:** 128K tokens
- **License:** Apache 2.0
- **Q4_K_M:** 7 GB ✅ (tight but manageable)
- **Link:** https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407
- **Features:** Tekken tokenizer (30% more efficient for code), multilingual Mistral AI GitHub

Phi-4 (14B) - NEW

- **Parameters:** 14B

- **Context:** 16K tokens
- **License:** MIT
- **Release:** December 12, 2024
- **MATH:** 80.4% (outperforms GPT-4o on math)
- **Q3_K_M:** ~7 GB (aggressive quantization)
- **Link:** https://huggingface.co/microsoft/phi-4
- **Note:** Specialized for mathematical reasoning Microsoft Community Hub +2

RECOMMENDED MODEL COMBINATIONS FOR 8GB VRAM

Configuration 1: Single Powerful Agent

- **Qwen2.5-Coder-7B-Instruct @ Q5_K_M** (~5.7 GB)
- **Total:** ~7 GB with context buffer
- **Best for:** Complex reasoning, coding, coordination in one model

Configuration 2: Coordinator + Executor

- **Coordinator:** Phi-3 Mini 3.8B @ Q8_0 (~4 GB) - planning/decomposition
- **Executor:** Llama 3.2 3B @ Q8_0 (~3.5 GB) - tool calling/execution
- **Total:** ~7.5 GB

Configuration 3: Multiple Lightweight Agents

- **3x Llama 3.2 1B @ Q8_0** (~6 GB total)
- OR **2x Llama 3.2 3B @ Q8_0** (~7 GB total)
- **Best for:** Distributed multi-agent systems

Configuration 4: Code-Focused Setup

- **Code:** Qwen2.5-Coder-7B @ Q4 (~3.8 GB)
- **Reasoning:** Llama 3.2 3B @ Q8 (~3.5 GB)
- **Total:** ~7.3 GB

PART 2: PACKAGES AND FRAMEWORKS

🚀 INFERENCE FRAMEWORKS

vLLM ⭐ RECOMMENDED FOR PRODUCTION

- **Latest Version:** 0.11.0 (October 2025), v1.0 alpha announced PyPI
- **Installation:** pip install vllm vLLM
- **GitHub:** https://github.com/vllm-project/vllm
- **PyPI:** https://pypi.org/project/vllm/
- **Documentation:** https://docs.vllm.ai/ vLLM

Key Features:

- PagedAttention for efficient memory management github
- GPTQ, AWQ, FP8, INT4/INT8 quantization support
- Continuous batching for high throughput github
- FlashAttention integration github vLLM
- RTX 4070 fully compatible (Compute Capability 8.9)
- Prefix caching, tensor parallelism github

TensorRT-LLM (NVIDIA)

- **Latest Version:** 1.1+ GitHub
- **Installation:** pip3 install tensorrt_llm GitHub
- **GitHub:** https://github.com/NVIDIA/TensorRT-LLM
- **Documentation:** https://nvidia.github.io/TensorRT-LLM/
- **Developer Site:** https://developer.nvidia.com/tensorrt

Key Features:

- Custom attention kernels optimized for NVIDIA GPUs
- FP8, FP4, INT4 AWQ, INT8 quantization
- Inflight batching, paged KV caching
- Speculative decoding
- RTX 4070 Ada Lovelace support

---

llama.cpp ⭐ BEST FOR GGUF

- **Installation:**
  - Homebrew: brew install llama.cpp
  - Source: git clone https://github.com/ggml-org/llama.cpp Hugging Face
- **GitHub:** https://github.com/ggml-org/llama.cpp
- **Documentation:** https://github.com/ggml-org/llama.cpp/blob/master/README.md
- **HuggingFace Guide:** https://huggingface.co/docs/hub/en/gguf-llamacpp

Key Features:

- Pure C/C++ implementation, minimal dependencies
- GGUF format support (1.5 to 8-bit quantization) PyPI
- GPU offloading with -ngl flag PyPI
- OpenAI-compatible server mode PyPI
- Direct HuggingFace model download Hugging Face
- Custom CUDA kernels for RTX 4070

Build with CUDA:

```bash
cmake -B build -DGGML_CUDA=ON
cmake --build build --config Release
```

Hugging Face

ExLlamaV2

- **Latest Version:** 0.0.12+
- **Installation:** pip install exllamav2
- **GitHub:** https://github.com/turboderp-org/exllamav2
- **Documentation:** https://github.com/turboderp-org/exllamav2/wiki

Key Features:

- EXL2 format with flexible bit-per-weight (2-8 bits) PyPI
- Flash Attention 2.5.7+ support GitHub
- Paged attention, Q4 cache mode PyPI
- Very fast inference (up to 56 tokens/sec on T4) Towards Data Science

- RTX 4070 Ampere+ fully supported

SGLang - BEST FOR MULTI-AGENT

- **Latest Version:** v0.5.3+
- **Installation:** pip install "sglang[all]"
- **GitHub:** https://github.com/sgl-project/sglang
- **Documentation:** https://sgl-project.github.io/

Key Features:

- **RadixAttention** for KV cache reuse (up to 6.4x faster)
- Zero-overhead CPU scheduler GitHub
- Structured output generation GitHub
- Multi-modal support (vision-language) GitHub
- **Ideal for complex multi-turn conversations and agent workflows**
- OpenAI-compatible API
- Used by xAI, ByteDance, LinkedIn GitHub

Text-generation-webui (Oobabooga)

- **Latest Version:** v3.16 (October 2025) GitHub
- **GitHub:** https://github.com/oobabooga/text-generation-webui
- **Releases:** https://github.com/oobabooga/text-generation-webui/releases

Key Features:

- Web UI for running any LLM locally
- Multiple backends: llama.cpp, ExLlamaV2, Transformers Libraries.io Internet Archive
- No-installation portable builds
- OpenAI-compatible API server GitHub
- Extensions for TTS, translation

Installation: Download one-click installer or:

```
bash
```

```
git clone https://github.com/oobabooga/text-generation-webui
cd text-generation-webui
./start_linux.sh  # or start_windows.bat
```

Oobabooga

🔧 QUANTIZATION TOOLS

AutoGPTQ

- **Latest Version:** 0.7.1 PyPI Stats
- **Installation:** pip install auto-gptq Libraries.io
- **GitHub:** https://github.com/AutoGPTQ/AutoGPTQ
- **PyPI:** https://pypi.org/project/auto-gptq/
- **Documentation:** https://autogptq.github.io/AutoGPTQ/

Key Features:

- 4-bit GPTQ quantization
- Marlin int4*fp16 kernel for Ampere GPUs Libraries.io
- ExLlamaV2 kernel by default
- Group sizes: 32, 64, 128
- Transformers integration
- RTX 4070 fully supported

Usage:

```python
from auto_gptq import AutoGPTQForCausalLM
model = AutoGPTQForCausalLM.from_quantized(
    "model_path",
    device="cuda:0",
    use_marlin=True  # For Ampere+
)
```

AutoAWQ

- **Latest Version:** 0.2.9 (⚠️ Deprecated as of 2025)

- **Installation:** pip install autoawq autoawq-kernels
- **GitHub:** https://github.com/casper-hansen/AutoAWQ (archived)
- **PyPI:** https://pypi.org/project/autoawq/
- **Documentation:** https://casper-hansen.github.io/AutoAWQ/

Key Features:

- 4-bit activation-aware weight quantization PyPI
- 3x speedup and 3x memory reduction PyPI
- ExLlamaV2 kernels support PyPI
- GGUF export capability PyPI
- RTX 4070 supported (Compute Capability 7.5+) GitHub Pydigger

Note: Officially deprecated May 2025, but existing versions work. PyPI

---

bitsandbytes ⭐ BEST FOR TRANSFORMERS

- **Latest Version:** 0.48.1 PyPI
- **Installation:** pip install bitsandbytes
- **GitHub:** https://github.com/bitsandbytes-foundation/bitsandbytes
- **PyPI:** https://pypi.org/project/bitsandbytes/
- **Documentation:** https://huggingface.co/docs/bitsandbytes/

Key Features:

- 8-bit and 4-bit quantization (LLM.int8()) GitHub
- QLoRA for parameter-efficient fine-tuning
- 8-bit optimizers GitHub for training
- Integrated with Transformers and Accelerate PyPI
- Load models in 4/8-bit directly from HuggingFace

Usage:

```
python
```

```
from transformers import AutoModelForCausalLM
model = AutoModelForCausalLM.from_pretrained(
    "model_name",
    load_in_4bit=True,
    device_map="auto"
)
```

PyPI

GGUF Conversion Tools

- **Repository:** https://github.com/ggml-org/llama.cpp
- **HuggingFace Tools:** https://huggingface.co/spaces/ggml-org/gguf-my-repo
- **Conversion Tutorial:** https://github.com/ggml-org/llama.cpp/discussions/7927

Conversion Process:

```bash
# Convert HuggingFace to GGUF
python convert_hf_to_gguf.py /path/to/model --outfile model.gguf --outtype q8_0

# Quantize further
./llama-quantize model.gguf model-q4_k_m.gguf Q4_K_M
```

GitHub

Quantization Types for 8GB:

- **Q4_K_M** - 4-bit (recommended for 7B-13B)
- **Q5_K_M** - 5-bit (better quality if VRAM allows)
- **Q8_0** - 8-bit (intermediate format)
- **Q3_K_M** - 3-bit (for 30B-70B models) GitHub +2

🤖 AGENT FRAMEWORKS

LangGraph ⭐ MOST ROBUST

- **Latest Version:** 1.0.1 (October 20, 2025)
- **Installation:** pip install langgraph
- **GitHub:** https://github.com/langchain-ai/langgraph

- **PyPI:** https://pypi.org/project/langgraph/
- **Documentation:** https://langchain-ai.github.io/langgraph/

Key Features:

- Stateful, multi-actor applications with LLMs
- Cycles, controllability, persistence
- Built-in checkpointing for error recovery
- Human-in-the-loop workflows
- Streaming support (token-level)
- Time travel debugging
- LangSmith integration for observability PyPI +2

AutoGen (Microsoft)

- **Latest Version:** 0.4.x (v4 architecture)
- **Installation:**
  - `pip install autogen-agentchat` (v0.4)
  - `pip install "autogen-agentchat" "autogen-ext[openai]"`
- **GitHub:** https://github.com/microsoft/autogen
- **PyPI:** https://pypi.org/project/autogen-agentchat/
- **Documentation:** https://microsoft.github.io/autogen/

Key Features:

- Event-driven, distributed, scalable multi-agent systems GitHub
- AgentChat API for rapid prototyping GitHub
- Core API for low-level control GitHub
- AutoGen Studio (no-code GUI) GitHub
- AutoGen Bench for evaluation GitHub
- **MCP (Model Context Protocol) integration**
- Magentic-One multi-agent system GitHub GitHub

Note: Package name changed from `pyautogen` to `autogen-agentchat` as of v0.2.36 PyPI

CrewAI ⭐ SIMPLEST TO USE

- **Installation:** pip install crewai crewai-tools
- **GitHub:** https://github.com/crewAIInc/crewAI
- **PyPI:** https://pypi.org/project/crewai/
- **Documentation:** https://docs.crewai.com/

Key Features:

- Role-based agent collaboration
- Built entirely from scratch (no LangChain dependency)
- Crews (autonomous) and Flows (event-driven)
- Deep customization at all levels
- Supports local LLMs (Ollama, LM Studio)
- 100,000+ certified developers
- MCP support GitHub +2

Integrations:

- Composio for 250+ tool integrations
- Qdrant, Weaviate, MongoDB for vector search

PydanticAI - BEST TYPE SAFETY

- **Installation:**
  - pip install pydantic-ai (full)
  - pip install pydantic-ai-slim (minimal)
- **GitHub:** https://github.com/pydantic/pydantic-ai
- **PyPI:** https://pypi.org/project/pydantic-ai/
- **Documentation:** https://ai.pydantic.dev/

Key Features:

- Type safety and validation powered by Pydantic GitHub
- Model-agnostic (20+ providers) GitHub
- Dependency injection for tools GitHub
- Logfire integration for observability GitHub
- Control flow via vanilla Python
- Structured result validation

MetaGPT - BEST FOR SOFTWARE ENGINEERING

- **Installation:** pip install --upgrade metagpt
- **GitHub:** https://github.com/FoundationAgents/MetaGPT
- **Documentation:** https://docs.deepwisdom.ai/

Key Features:

- Simulates software company (PM, Architect, Engineer, QA)
- Encodes Standardized Operating Procedures (SOPs)
- Assembly line paradigm for task decomposition
- Takes one-line requirement as input
- Structured outputs (PRD, design docs, code, tests)
- ICLR 2024 oral presentation (top 1.2%) GitHub +3

AgentOps - MONITORING

- **Installation:** pip install agentops
- **GitHub:** https://github.com/AgentOps-AI/agentops
- **PyPI:** https://pypi.org/project/agentops/
- **Website:** https://www.agentops.ai/

Key Features:

- Comprehensive agent observability
- Real-time monitoring with session replays
- LLM cost tracking and budgeting
- Failure detection for multi-agent systems
- Works with 400+ LLMs

Integrations: CrewAI (2-line), AutoGen, LangChain, OpenAI Agents SDK GitHub GitHub

---

🧪 **EVALUATION TOOLS**

SWE-bench ⭐ STANDARD BENCHMARK

- **Installation:**

```bash
git clone https://github.com/princeton-nlp/SWE-bench.git
cd SWE-bench
pip install -e .
```

- **GitHub:** https://github.com/SWE-bench/SWE-bench
- **Website:** https://www.swebench.com/
- **Dataset:** from datasets import load_dataset; swebench = load_dataset('princeton-nlp/SWE-bench', split='test')

Key Features:

- 2,294 real-world GitHub issues from Python repos Deepeval +3
- **SWE-bench Lite:** 300 instances subset
- **SWE-bench Verified:** 500 human-validated (OpenAI collaboration) DEV Community
- Docker-based evaluation harness GitHub
- Deterministic unit test evaluation GitHub +4

Evaluation Command:

```bash
python -m swebench.harness.run_evaluation \
  --dataset_name princeton-nlp/SWE-bench_Lite \
  --predictions_path <path> \
  --max_workers <num> \
  --run_id <id>
```

System Requirements: 120GB storage, 16GB RAM, 8 CPU cores, Docker GitHub GitHub

---

HumanEval

- **Installation:**

```bash
git clone https://github.com/openai/human-eval
pip install -e human-eval
```

- **GitHub:** https://github.com/openai/human-eval
- **Paper:** "Evaluating Large Language Models Trained on Code" (OpenAI)

Key Features:

- 164 hand-crafted Python programming problems Deepeval
- Function-level code generation
- pass@k metric Klu (k=1, 10, 100)
- Execution-based verification Cerebras Medium

Evaluation:

```
bash

evaluate_functional_correctness <samples.jsonl>
```

Variants:

- **HumanEval+:** Extended test cases
- **HumanEval-X:** Multi-language (C++, Java, JS, Go)
- **HumanEval-V:** Visual reasoning (253 tasks)

E2B (Code Execution Sandbox)

- **Installation:**
  - Python: pip install e2b-code-interpreter
  - JavaScript: npm install @e2b/code-interpreter
- **GitHub:** https://github.com/e2b-dev/E2B
- **Documentation:** https://e2b.dev/docs
- **Website:** https://e2b.dev/

Key Features:

- Sandboxes start in ~150ms
- Firecracker microVMs for isolation Walturn
- Up to 24-hour runtime Walturn
- Filesystem I/O, package installation on-the-fly AI Agents Directory
- Self-hosting available (Terraform: GCP, AWS) Walturn

Python Example:

```
python
```

```
from e2b_code_interpreter import Sandbox
with Sandbox() as sandbox:
    execution = sandbox.run_code("x = 1 + 1; x")
    print(execution.text)  # outputs 2
```

Modal (Serverless Compute)

- **Website:** https://modal.com/
- **Documentation:** https://modal.com/docs

Key Features:

- Fast sandbox boot (<90ms)
- gVisor-based containerization
- GPU support available
- Horizontal scaling via API
- Built-in Git and LSP support

Note: Fully managed platform (no self-hosting)

🧠 MEMORY / VECTOR STORES

ChromaDB ⭐ SIMPLEST

- **Latest Version:** 1.2.2
- **Installation:** `pip install chromadb`
- **GitHub:** https://github.com/chroma-core/chroma
- **PyPI:** https://pypi.org/project/chromadb/
- **Documentation:** https://docs.trychroma.com/

Key Features:

- In-memory or persistent storage
- Built-in embedding (Sentence Transformers)
- Metadata filtering
- Multi-modal support (text, images)
- Chroma Cloud (hosted SaaS) PyPI +3

Usage:

```python
import chromadb
client = chromadb.Client()  # in-memory
collection = client.create_collection("my_collection")
collection.add(
    documents=["doc1", "doc2"],
    metadatas=[{"source": "web"}, {"source": "pdf"}],
    ids=["id1", "id2"]
)
results = collection.query(query_texts=["query"], n_results=2)
```

FAISS ⭐ BEST FOR SCALE

- **Latest Version:** 1.12.0 (August 2025)
- **Installation:**
  - CPU: `pip install faiss-cpu`
  - GPU: `pip install faiss-gpu`
- **GitHub:** https://github.com/facebookresearch/faiss
- **PyPI:** https://pypi.org/project/faiss-cpu/
- **Documentation:** https://faiss.ai/

Key Features:

- Handles billion-scale datasets
- Multiple indexing (IVF, HNSW, PQ)
- GPU acceleration support
- L2 and dot product distances
- Compression for large datasets

Usage:

```python
```

```
import faiss
import numpy as np
d = 128  # dimension
index = faiss.IndexFlatL2(d)
vectors = np.random.random((1000, d)).astype('float32')
index.add(vectors)
distances, indices = index.search(query, k=5)
```

Qdrant - BEST FILTERING

- **Installation:** `pip install qdrant-client`
- **GitHub:** https://github.com/qdrant/qdrant
- **PyPI:** https://pypi.org/project/qdrant-client/
- **Documentation:** https://qdrant.tech/documentation/

Key Features:

- In-memory, local disk, or cloud deployment
- **Advanced filtering with metadata**
- Multiple distance metrics (Cosine, Euclidean, Dot)
- Payload-based filtering
- gRPC and REST APIs
- Horizontal scaling
- FastEmbed inference API

Usage:

```python
from qdrant_client import QdrantClient
client = QdrantClient(":memory:")  # or path/URL
client.create_collection(
    collection_name="my_collection",
    vectors_config=VectorParams(size=128, distance=Distance.COSINE)
)
```

Weaviate - BEST FOR HYBRID SEARCH

- **Installation:** `pip install weaviate-client`

- **GitHub:** https://github.com/weaviate/weaviate
- **Documentation:** https://weaviate.io/developers/weaviate

Key Features:

- Vector similarity, full-text, hybrid search
- Built-in vectorization modules (text2vec, multi2vec)
- GraphQL and REST APIs
- Multimodal data support
- Generative search (RAG)
- Horizontal scaling

LanceDB - BEST FOR EMBEDDED

- **Installation:** `pip install lancedb`
- **GitHub:** https://github.com/lancedb/lancedb
- **Documentation:** https://lancedb.github.io/lancedb/

Key Features:

- Fast vector search (billions of vectors)
- **Embedded and serverless** (no separate server)
- SQL support alongside vector search
- Multimodal (text, images, videos)
- Zero-copy, automatic versioning
- GPU support for indexing

Usage:

```
python
```

```
import lancedb
db = lancedb.connect("./my_db")
table = db.create_table(
    "vectors",
    data=[
        {"vector": [1.1, 1.2], "text": "doc1"},
        {"vector": [0.2, 1.8], "text": "doc2"}
    ]
)
results = table.search([1.0, 1.0]).limit(5).to_list()
```

📡 MODEL CONTEXT PROTOCOL (MCP)

MCP Python SDK

- **Latest Version:** 1.19.0 (October 24, 2025)
- **Installation:** `pip install "mcp[cli]"` or `uv add "mcp[cli]"`
- **GitHub:** https://github.com/modelcontextprotocol/python-sdk
- **PyPI:** https://pypi.org/project/mcp/
- **Documentation:** https://modelcontextprotocol.io/

Key Features:

- Full MCP specification implementation
- Build MCP clients and servers
- Standard transports: stdio, SSE, Streamable HTTP
- FastMCP high-level interface
- OAuth 2.1 support
- Structured output with Pydantic

SDK Structure:

- `mcp.server.fastmcp`: High-level framework
- `mcp.server.lowlevel`: Low-level implementation
- `mcp.client`: Client session management
- `mcp.server.auth`: Authentication

MCP TypeScript SDK

- **Latest Version:** 1.20.2 (4 days ago as of October 2025)
- **Installation:** `npm install @modelcontextprotocol/sdk`
- **GitHub:** https://github.com/modelcontextprotocol/typescript-sdk
- **npm:** https://www.npmjs.com/package/@modelcontextprotocol/sdk
- **Documentation:** https://modelcontextprotocol.io/

Key Features:

- Full MCP specification
- Type-safe tool definitions with Zod schemas
- Resource templates
- Streamable HTTP transport
- Node.js v18+ required

Modules:

- `@modelcontextprotocol/sdk/server/mcp.js`
- `@modelcontextprotocol/sdk/server/stdio.js`
- `@modelcontextprotocol/sdk/server/streamableHttp.js`
- `@modelcontextprotocol/sdk/client`

MCP Official Examples

- **Repository:** https://github.com/modelcontextprotocol/servers
- **Awesome MCP Servers:**
  - https://github.com/punkpeye/awesome-mcp-servers
  - https://github.com/wong2/awesome-mcp-servers

Reference Servers:

- **Everything:** Reference/test server
- **Fetch:** Web content fetching
- **Filesystem:** Secure file operations
- **Git:** Repository manipulation
- **Memory:** Knowledge graph-based memory
- **Sequential Thinking:** Dynamic problem-solving

Community Servers:

- GitHub: https://github.com/github/github-mcp-server
- Google Drive, Slack, Postgres, Puppeteer, Stripe

Tutorials:

- Anthropic Course: https://anthropic.skilljar.com/introduction-to-model-context-protocol
- FreeCodeCamp: https://www.freecodecamp.org/news/how-to-build-a-custom-mcp-server-with-typescript-a-handbook-for-developers/
- Microsoft: https://github.com/microsoft/mcp-for-beginners

PART 3: COMPREHENSIVE SOURCE LINKS

📚 ARXIV PAPERS

Software Engineering Agents

- **SWE-agent:** https://arxiv.org/abs/2405.15793
- **SWE-bench (original):** https://arxiv.org/abs/2310.06770
- **SWE-bench Pro:** https://arxiv.org/abs/2509.16941
- **SWE-Search (MCTS):** https://arxiv.org/abs/2410.20285
- **DEI (Diversity):** https://arxiv.org/abs/2408.07060
- **RL Training:** https://arxiv.org/abs/2508.03501
- **SWE-Dev:** https://arxiv.org/abs/2506.07636
- **ToM-SWE:** https://arxiv.org/abs/2510.21903

Code Generation Agents

- **Survey on Code Generation:** https://arxiv.org/abs/2508.00083
- **Self-Organized Agents:** https://arxiv.org/abs/2404.02183
- **Experience Replay:** https://arxiv.org/abs/2410.12236
- **CodeAgent:** https://arxiv.org/abs/2401.07339
- **Guided Code Generation:** https://arxiv.org/abs/2501.06625
- **Quantum Code Generation:** https://arxiv.org/abs/2504.14557

Multi-Agent Systems & Benchmarks

- **SwarmBench:** https://arxiv.org/abs/2505.04364
- **SwarmAgentic:** https://arxiv.org/abs/2506.15672

- **MultiAgentBench:** https://arxiv.org/abs/2503.01935
- **BenchAgents:** https://arxiv.org/abs/2410.22584
- **Multi-Agent Social Science:** https://arxiv.org/abs/2506.01839
- **Multi-Agent Specification:** https://arxiv.org/abs/2506.10467

Model Papers

- **Qwen2.5-Coder:** https://arxiv.org/abs/2409.12186
- **DeepSeek-Coder-V2:** https://arxiv.org/abs/2406.11931
- **StarCoder2:** https://arxiv.org/abs/2402.19173
- **Phi-3:** https://arxiv.org/pdf/2404.14219
- **Gemma 2:** https://arxiv.org/pdf/2408.00118

💻 GITHUB REPOSITORIES

MCP Official

- **Specification:** https://github.com/modelcontextprotocol/modelcontextprotocol
- **Python SDK:** https://github.com/modelcontextprotocol/python-sdk
- **TypeScript SDK:** https://github.com/modelcontextprotocol/typescript-sdk
- **Servers:** https://github.com/modelcontextprotocol/servers
- **Organization:** https://github.com/modelcontextprotocol

MCP Community

- **GitHub MCP Server:** https://github.com/github/github-mcp-server
- **Awesome MCP (punkpeye):** https://github.com/punkpeye/awesome-mcp-servers
- **Awesome MCP (wong2):** https://github.com/wong2/awesome-mcp-servers
- **Simple Example:** https://github.com/alejandro-ao/mcp-server-example
- **Microsoft Beginners:** https://github.com/microsoft/mcp-for-beginners
- **Quick Example:** https://github.com/ALucek/quick-mcp-example
- **FastMCP (Python):** https://github.com/jlowin/fastmcp
- **FastMCP (TypeScript):** https://github.com/punkpeye/fastmcp
- **MCP-USE:** https://github.com/mcp-use/mcp-use-ts

Inference & Quantization

- **vLLM:** https://github.com/vllm-project/vllm

- **TensorRT-LLM:** https://github.com/NVIDIA/TensorRT-LLM
- **llama.cpp:** https://github.com/ggml-org/llama.cpp
- **ExLlamaV2:** https://github.com/turboderp-org/exllamav2
- **SGLang:** https://github.com/sgl-project/sglang
- **Text-generation-webui:** https://github.com/oobabooga/text-generation-webui
- **AutoGPTQ:** https://github.com/AutoGPTQ/AutoGPTQ
- **AutoAWQ:** https://github.com/casper-hansen/AutoAWQ
- **bitsandbytes:** https://github.com/bitsandbytes-foundation/bitsandbytes

Agent Frameworks

- **LangGraph:** https://github.com/langchain-ai/langgraph
- **AutoGen:** https://github.com/microsoft/autogen
- **CrewAI:** https://github.com/crewAIInc/crewAI
- **PydanticAI:** https://github.com/pydantic/pydantic-ai
- **MetaGPT:** https://github.com/FoundationAgents/MetaGPT
- **AgentOps:** https://github.com/AgentOps-AI/agentops

Benchmarks

- **SWE-bench:** https://github.com/SWE-bench/SWE-bench
- **SWE-agent:** https://github.com/SWE-agent/SWE-agent
- **SWE-PolyBench:** https://github.com/amazon-science/SWE-PolyBench
- **SwarmBench:** https://github.com/RUC-GSAI/YuLan-SwarmIntell
- **HumanEval:** https://github.com/openai/human-eval
- **E2B:** https://github.com/e2b-dev/E2B

Vector Stores

- **ChromaDB:** https://github.com/chroma-core/chroma
- **FAISS:** https://github.com/facebookresearch/faiss
- **Qdrant:** https://github.com/qdrant/qdrant
- **Weaviate:** https://github.com/weaviate/weaviate
- **LanceDB:** https://github.com/lancedb/lancedb

OpenAI Swarm

- **OpenAI Swarm:** https://github.com/openai/swarm
- **Agency Swarm:** https://github.com/VRSEN/agency-swarm

- **Swarms (Production):** https://github.com/kyegomez/swarms
- **OpenAI Agent Swarm:** https://github.com/daveshap/OpenAI_Agent_Swarm
- **Open-Swarm:** https://github.com/marcusschiesser/open-swarm

Code Generation Research

- **Awesome LLM Agent4Code:** https://github.com/JiaruQian/awesome-llm-based-agent4code

---

📖 DOCUMENTATION SITES

MCP

- **Main Site:** https://modelcontextprotocol.io/
- **Examples:** https://modelcontextprotocol.io/examples
- **Anthropic Course:** https://anthropic.skilljar.com/introduction-to-model-context-protocol

Inference Frameworks

- **vLLM Docs:** https://docs.vllm.ai/
- **TensorRT-LLM:** https://nvidia.github.io/TensorRT-LLM/
- **TensorRT Developer:** https://developer.nvidia.com/tensorrt
- **llama.cpp Guide:** https://huggingface.co/docs/hub/en/gguf-llamacpp
- **SGLang:** https://sgl-project.github.io/

Quantization

- **AutoGPTQ:** https://autogptq.github.io/AutoGPTQ/
- **AutoAWQ:** https://casper-hansen.github.io/AutoAWQ/
- **bitsandbytes:** https://huggingface.co/docs/bitsandbytes/

Agent Frameworks

- **LangGraph:** https://langchain-ai.github.io/langgraph/
- **AutoGen:** https://microsoft.github.io/autogen/
- **CrewAI:** https://docs.crewai.com/
- **PydanticAI:** https://ai.pydantic.dev/
- **MetaGPT:** https://docs.deepwisdom.ai/
- **AgentOps:** https://www.agentops.ai/

OpenAI

- **Agents SDK MCP:** https://openai.github.io/openai-agents-python/mcp/

Anthropic

- **Claude API:** https://www.anthropic.com/api
- **Claude Docs:** https://docs.claude.com/
- **Build with Claude:** https://www.anthropic.com/learn/build-with-claude
- **MCP Announcement:** https://www.anthropic.com/news/model-context-protocol

Vector Stores

- **ChromaDB:** https://docs.trychroma.com/
- **FAISS:** https://faiss.ai/
- **Qdrant:** https://qdrant.tech/documentation/
- **Weaviate:** https://weaviate.io/developers/weaviate
- **LanceDB:** https://lancedb.github.io/lancedb/

Sandboxes

- **E2B:** https://e2b.dev/docs
- **Modal:** https://modal.com/docs

AWS

- **Bedrock Messages:** https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-anthropic-claude-messages.html
- **Tool Use:** https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-anthropic-claude-messages-tool-use.html

---

📦 PACKAGE REGISTRIES

PyPI (Python)

- **mcp:** https://pypi.org/project/mcp/
- **vllm:** https://pypi.org/project/vllm/
- **langgraph:** https://pypi.org/project/langgraph/
- **autogen-agentchat:** https://pypi.org/project/autogen-agentchat/
- **crewai:** https://pypi.org/project/crewai/

- **pydantic-ai:** https://pypi.org/project/pydantic-ai/
- **agentops:** https://pypi.org/project/agentops/
- **auto-gptq:** https://pypi.org/project/auto-gptq/
- **autoawq:** https://pypi.org/project/autoawq/
- **bitsandbytes:** https://pypi.org/project/bitsandbytes/
- **chromadb:** https://pypi.org/project/chromadb/
- **faiss-cpu:** https://pypi.org/project/faiss-cpu/
- **qdrant-client:** https://pypi.org/project/qdrant-client/
- **weaviate-client:** https://pypi.org/project/weaviate-client/

npm (JavaScript/TypeScript)

- **@modelcontextprotocol/sdk:** https://www.npmjs.com/package/@modelcontextprotocol/sdk

---

🏆 BENCHMARK WEBSITES

- **SWE-bench:** https://www.swebench.com/
- **SWE-bench Original:** https://www.swebench.com/original.html
- **EvalPlus:** https://evalplus.github.io/leaderboard.html

📝 TECHNICAL BLOG POSTS

MCP

- **Descope (What is MCP):** https://www.descope.com/learn/post/mcp
- **KeywordsAI Guide:** https://www.keywordsai.co/blog/introduction-to-mcp
- **Medium Deep Dive:** https://medium.com/@amanatulla1606/anthropics-model-context-protocol-mcp-a-deep-dive-for-developers-1d3db39c9fdc
- **GitHub MCP Walkthrough:** https://sagarag.medium.com/unwrapping-mcp-a-walkthrough-with-the-github-mcp-server-293608deaec8
- **GitHub Blog (Practical Guide):** https://github.blog/ai-and-ml/generative-ai/a-practical-guide-on-how-to-use-the-github-mcp-server/
- **FreeCodeCamp Tutorial:** https://www.freecodecamp.org/news/how-to-build-a-custom-mcp-server-with-typescript-a-handbook-for-developers/

Multi-Agent Systems

- **LangChain Benchmarking:** https://blog.langchain.com/benchmarking-multi-agent-architectures/
- **OpenAI Swarm Exploration:** https://medium.com/@michael_79773/exploring-openais-swarm-an-experimental-framework-for-multi-agent-systems-5ba09964ca18
- **FutureSmart AI Swarm:** https://blog.futuresmart.ai/openai-swarm-a-hands-on-introduction

SWE-bench

- **Princeton Blog:** https://pli.princeton.edu/blog/2023/swe-bench-can-language-models-resolve-real-world-github-issues
- **Cognition AI (Devin):** https://cognition.ai/blog/swe-bench-technical-report
- **Dev.to Guide:** https://dev.to/duplys/swe-bench-swe-bench-verified-benchmarks-1cm
- **OpenAI (Verified):** https://openai.com/index/introducing-swe-bench-verified/

Model Announcements

- **Qwen Blog:** https://qwenlm.github.io/blog/qwen2.5-coder-family/
- **StarCoder2:** https://huggingface.co/blog/starcoder2
- **Llama 3.1:** https://ai.meta.com/blog/meta-llama-3-1/
- **Llama 3.2:** https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/
- **Phi-3:** https://azure.microsoft.com/en-us/blog/introducing-phi-3-redefining-whats-possible-with-slms/
- **Phi-4:** https://techcommunity.microsoft.com/blog/azure-ai-foundry-blog/introducing-phi-4-microsoft%E2%80%99s-newest-small-language-model-specializing-in-comple/4357090
- **Gemma 2:** https://blog.google/technology/developers/google-gemma-2/
- **Mistral NeMo:** https://mistral.ai/news/mistral-nemo

Other Resources

- **Zapier Claude API:** https://zapier.com/blog/claude-api/
- **OpenAI Community Swarm:** https://community.openai.com/t/openai-swarm-for-agents-and-agent-handoffs/976579

🎯 MODEL HUB ORGANIZATIONS

- **Qwen:** https://huggingface.co/Qwen
- **DeepSeek-AI:** https://huggingface.co/deepseek-ai
- **BigCode:** https://huggingface.co/bigcode
- **CodeLlama (Meta):** https://huggingface.co/codellama
- **Meta (Llama):** https://huggingface.co/meta-llama

- **Microsoft (Phi):** https://huggingface.co/microsoft
- **Google (Gemma):** https://huggingface.co/google
- **Mistral:** https://huggingface.co/mistralai

SUMMARY & RECOMMENDATIONS

🎯 Quick Start: Best Stack for RTX 4070 8GB

For Code Generation:

- **Model:** Qwen2.5-Coder-7B-Instruct @ Q5_K_M (5.7 GB)
- **Inference:** vLLM or llama.cpp
- **Quantization:** Q5_K_M via GGUF

For Multi-Agent Coordination:

- **Framework:** LangGraph or CrewAI
- **MCP:** Python SDK 1.19.0
- **Monitoring:** AgentOps
- **Memory:** ChromaDB or Qdrant

For Evaluation:

- **Benchmarks:** SWE-bench Lite, HumanEval
- **Sandbox:** E2B
- **Vector Store:** ChromaDB (simplest) or FAISS (fastest)

Optimal Configuration:

```bash
# Install core stack
pip install vllm langgraph chromadb "mcp[cli]" agentops

# Download model
huggingface-cli download Qwen/Qwen2.5-Coder-7B-Instruct-GGUF \
  qwen2.5-coder-7b-instruct-q5_k_m.gguf --local-dir ./models

# Run with llama.cpp
llama-server -m ./models/qwen2.5-coder-7b-instruct-q5_k_m.gguf -ngl 40
```

🏆 Performance Rankings

Code Generation (HumanEval):

1. Qwen2.5-Coder-7B-Instruct: **84.1%**
2. DeepSeek-Coder-V2-Lite-Instruct: **81.1%**
3. Llama 3.1 8B Instruct: **72.6%**
4. Phi-3 Mini: **59-61%**
5. CodeLlama-7B-Instruct: **50.6%**
6. StarCoder2-7B: **35.4%**

Reasoning (MMLU):

1. Phi-3 Small 7B: **75.3%**
2. Llama 3.1 8B: **69.4%**
3. Phi-3 Mini: **68.8%**
4. Llama 3.2 3B: **63.4%**

💡 Key Insights

1. **Qwen2.5-Coder-7B dominates** code generation benchmarks (84.1% HumanEval) while fitting comfortably in 8GB VRAM with Q4/Q5 quantization
2. **MCP adoption is rapid** - OpenAI integrated it across ChatGPT, Agents SDK in March 2025; perfect for multi-agent tool use
3. **Multi-agent benefits are real** - Research shows different agents solve different problems (54.3% oracle coverage vs 26.6% single agent)
4. **RL doubles performance** - Training with RL improves SWE-bench from 20% → 39%
5. **SGLang is ideal for agents** - RadixAttention provides 6.4x speedup for multi-turn conversations
6. **ChromaDB is easiest** for prototyping, **FAISS for scale**, **Qdrant for filtering**

MEMORY FOOTPRINT REFERENCE

Model	Parameters	Q4_0	Q5_0	Q8_0	Fits 8GB?
Qwen2.5-Coder-7B	7.61B	3.8 GB	4.75 GB	7.61 GB	✅ All
Llama 3.1 8B	8B	4.9 GB	5.7 GB	6.6 GB	✅ All
StarCoder2-7B	7B	4.0 GB	4.94 GB	7.63 GB	✅ All
Gemma 2 9B	9B	5.5 GB	6.5 GB	8.5 GB	⚠️ Q4/Q5 only

Model	Parameters	Q4_0	Q5_0	Q8_0	Fits 8GB?
Mistral NeMo 12B	12B	7.0 GB	8+ GB	10+ GB	⚠️ Q4 only
Phi-3 Mini	3.8B	2.5 GB	3.5 GB	4 GB	✅ All
Llama 3.2 3B	3B	2 GB	2.5 GB	3.5 GB	✅ All
Llama 3.2 1B	1B	1 GB	1.3 GB	1.8 GB	✅ All

Remember: Add 1-2 GB for context/KV cache when calculating total VRAM usage.

Report compiled: October 28, 2025
Total resources: 100+ links across models, frameworks, papers, and tools
Focus: Production-ready multi-agent software engineering on consumer hardware