# 本地化神经符号AI小说创作系统架构与实施深度研究报告

## 1. 执行摘要与架构哲学

随着大语言模型（LLM）能力的飞跃，创意写作领域正经历一场从辅助生成向\*\*认知架构（Cognitive Architecture）**转型的技术革命。本报告旨在深度剖析并构建一套基于VSCode环境、以本地模型（如Codex/Llama）为驱动核心的AI长篇小说自动化写作系统。该系统的核心设计理念在于克服当前LLM在长文本生成中普遍存在的“灾难性遗忘”与“逻辑幻觉”问题，通过引入**神经符号（Neuro-Symbolic）\*\*方法，将神经网络的生成流动性与知识图谱（Knowledge Graph）的逻辑刚性相结合，实现百万字量级长篇叙事的连贯性与深度1。

传统的AI写作工具往往仅扮演“随机鹦鹉”的角色，基于有限的上下文窗口预测下一个Token。这种模式在短篇创作中尚可，但在构建庞大的世界观与复杂的人物弧光时，必然面临前后设定冲突、伏笔遗失等问题。本系统基于METATRON框架的设计思想，将创作过程解构为\*\*规划（符号/认知层）**与**撰写（神经/生成层）\*\*两个独立的维度1。LLM在此架构中不再仅仅是生成器，而是升级为调度中央处理器（CPU），负责指挥各类专用智能体（Agents）调用外部存储模块（RAM/HDD）。

选择VSCode作为集成开发环境（IDE），实质上是将小说创作重构为**软件开发生命周期（SDLC）**。每一章节被视为功能模块，人物设定被视为类与对象，剧情分支则利用Git进行版本控制与回滚。这种工程化的视角赋予了创作者对文本极细颗粒度的控制权，允许通过Diff工具对比AI生成的草稿与润色稿，实现人机协同的深度耦合1。此外，全本地化的运行环境（Localhost Deployment）通过Docker容器化部署Neo4j图数据库与ChromaDB向量数据库，不仅消除了对昂贵在线API的依赖，更构建了数据主权与隐私保护的“护城河”，允许创作者在无审查干扰的环境下探索深度叙事主题3。

## 2. 认知智能体工作流：从突触到叙事

本系统的“大脑”并非单一模型，而是一个由多个专用智能体组成的协作网络。这种\*\*多智能体协作机制（Multi-Agent Collaboration Mechanism）\*\*通过分工明确的流水线，有效降低了单个生成任务的认知负载，确保了叙事逻辑的严密性1。

### 2.1 概念规划智能体（The Architect）：顶层设计的逻辑引擎

“概念规划师”智能体位于创作链条的最上游，负责从零构建故事蓝图。它不直接参与正文撰写，而是专注于叙事结构的搭建。该智能体依据用户输入的题材（如“赛博朋克复仇剧”）与核心创意，调用预设的genre\_templates（题材模版），生成层级化的plot\_timeline（情节进度线）1。

这一过程不仅是简单的线性列表生成，而是涉及复杂的因果推理。规划代理需运用三幕剧或起承转合等经典叙事理论，规划出关键的情节转折点（Plot Points）。例如，在玄幻题材中，它需设计主角的升级路线与势力分布；在推理题材中，则需预埋案件线索与反转节点。其输出通常为结构化的JSON或YAML文件，详细定义了每一章的叙事目标、登场人物及核心冲突，为后续的撰写环节提供了不可逾越的逻辑约束1。

### 2.2 章节创作智能体（The Writer）：受控生成的执行终端

“章节创作师”是系统的核心生成引擎，其运作模式体现了“受控生成”的精髓。它并不自由发挥，而是严格遵循规划智能体制定的大纲指令。在生成每一章正文前，系统会通过RAG（检索增强生成）技术，从记忆库中提取与当前章节强相关的人物档案、环境描写及前情提要，拼接成涵盖全面上下文的Prompt输入给模型1。

为了解决单一生成结果质量不稳定的问题，该智能体采用了\*\*多版本并发生成（Multi-Drafting）\*\*策略。针对同一大纲节点，系统会通过调整Prompt中的随机种子或侧重点（如“侧重心理描写”vs“侧重动作场面”），并行生成多个版本的草稿（versions=3）。这种机制类似于集束搜索（Beam Search），允许后续的评价智能体从中优选出逻辑最通顺、文风最契合的版本，从而规避了单路径生成可能导致的死胡同1。

此外，该智能体具备**动态长度控制能力**。依据大纲中标记的“剧情强度”标签，智能体能够自动调整输出策略：对于标记为“高潮/大战”的章节，系统会自动提升max\_tokens参数，并注入“详尽描写环境细节、动作拆解”的指令，确保生成内容达到3200字以上的丰富度；而对于“过渡/日常”章节，则限制篇幅在2000字左右，强调叙事节奏的紧凑性。这种动态调节机制模拟了人类作家在不同情节段落间的笔力分配，有效提升了整部作品的阅读节奏感1。

### 2.3 内容压缩智能体（The Archivist）：长期记忆的守护者

随着故事推进，上下文窗口迅速耗尽是长篇写作面临的最大技术瓶颈。“内容压缩师”智能体正是为解决此问题而生。在每一章正文生成并定稿后，该智能体介入工作，对文本进行深度阅读理解与**信息抽取（Information Extraction, IE）**1。

其核心任务是将非结构化的自然语言文本转化为结构化的知识条目。例如，当文中提到“主角获得了一把名为‘断钢’的长剑，且该剑在月光下会发光”时，压缩智能体需识别出实体（Item: 断钢）、属性（Effect: 月光发光）及关系（Possess: 主角 -> 断钢），并更新至elements.json或Neo4j图数据库中。同时，它还需提炼出本章的剧情摘要，追加至plot\_timeline文件中。这一过程相当于将短时工作记忆“提交”至长时硬盘存储，确保系统在生成第五十章时，依然能准确调用第一章设定的伏笔，实现真正的“不遗忘”1。

### 2.4 评价与修缮智能体（The Critics）：质量控制闭环

为了确保输出内容的专业性，系统引入了类似编辑部的审核机制。

* **评价智能体（Evaluator）：** 扮演“毒舌编辑”的角色。它利用Codex模型强大的逻辑分析能力，对草稿进行多维度的审阅，包括逻辑一致性（如“角色A不可能同时出现在两地”）、人设符合度（“性格懦弱的角色突然暴怒缺乏铺垫”）及行文流畅度。其输出是一份结构化的评审报告，列出具体的修改建议与问题点1。
* **润色智能体（Editor）：** 则根据评审报告进行针对性的修订。它不仅仅是简单的语法纠错，更能执行风格统一化操作（如将网络用语替换为古风辞藻），甚至重写整个段落以修复逻辑漏洞。这一“生成-评价-修改”的闭环迭代，显著提升了AI生成文本的可读性与文学性1。
* **安全过滤智能体（Filter）：** 在本地部署敏感词库，对最终文本进行正则扫描与关键词过滤，确保内容符合特定的合规要求，且无需上传数据至云端API进行审查，保障了创作的私密性1。

## 3. 双重记忆系统：符号图谱与向量空间的融合

为了模拟人类极其复杂的记忆机制——既包含精确的事实记忆（语义记忆），也包含模糊的情感与场景记忆（情景记忆），本系统构建了\*\*混合RAG（Hybrid RAG）\*\*架构。这一架构巧妙地融合了知识图谱（Neo4j）的结构化优势与向量数据库（ChromaDB）的语义检索能力，为AI提供了全息的上下文支撑2。

### 3.1 符号记忆：Neo4j知识图谱的构建与应用

在小说创作中，人物关系、地理位置、物品归属等信息必须是确定性的。向量检索基于概率相似度，往往难以精确回答“谁是主角的父亲”这类硬性逻辑问题。因此，引入\*\*图数据库（Graph Database）\*\*作为“世界设定集（World Bible）”的存储载体至关重要6。

#### 3.1.1 适应虚构叙事的Schema设计

系统采用属性图（Property Graph）模型，将叙事元素抽象为节点（Nodes）与关系（Relationships）。与传统企业级知识图谱不同，小说创作的图谱Schema设计需更具灵活性与文学性2：

* **节点类型（Labels）：**
  + Character（角色）：属性包含name（姓名）、personality（性格标签）、appearance（外貌特征）、status（当前状态，如“存活/受伤”）。
  + Location（地点）：属性包含name、coordinates、atmosphere（氛围基调）、resources（特产）。
  + Event（事件）：属性包含timestamp、summary、consequence（后果）。
  + Item（物品/道具）：属性包含origin、function、owner。
* **关系类型（Relationships）：**
  + (:Character)-->(:Character)：复杂的人际关系网。
  + (:Character)-->(:Location)：追踪角色的时空轨迹。
  + (:Item)-->(:Character)：物品所有权的流转。
  + (:Event)-->(:Event)：事件间的因果链条。

这种结构化存储使得“写作智能体”能够执行极其复杂的逻辑查询。例如，当剧情需要安排一场冲突时，系统可以执行Cypher查询语句：“查找所有当前位于‘帝都’且与主角关系为‘仇恨’的角色”，Neo4j将返回精确的候选人名单，而非模糊的文本片段，从而确保剧情发展的逻辑严密性9。

#### 3.1.2 动态Schema演化与维护

小说创作是一个动态过程，新的概念（如某种新的魔法体系）可能随剧情发展而诞生。传统的数据库Schema往往僵化，难以适应这种变化。本系统利用LLM的推理能力，实现了Schema的动态推断与扩展。当“压缩智能体”在文中识别出一种前所未见的关系类型（如“灵魂绑定”）时，它会自动在Neo4j中创建新的关系类型，而非受限于预定义的模板。这种\*\*本体自动构建（Ontology Learning）\*\*机制，使得知识图谱能够伴随小说的成长而自我进化，始终保持对最新世界观的精准映射6。

### 3.2 语义记忆：ChromaDB向量库的沉浸式检索

虽然图谱擅长处理事实，但它无法捕捉“氛围”、“语气”或“文风”等非结构化信息。ChromaDB作为系统的“情景记忆（Episodic Memory）”模块，承担了存储叙事文本流的任务5。

#### 3.2.1 文本分块与向量化策略

系统将生成的小说正文按段落或场景进行分块（Chunking），并利用嵌入模型（Embedding Model，如本地的all-MiniLM-L6-v2）将其转化为高维向量存储于ChromaDB中。

* **文风一致性维持：** 当需要描写一个“悲伤的雨夜”时，系统可以检索向量库中语义相似的历史片段。通过参考之前的描写手法与用词习惯，AI能够模仿自身的文风，避免前后章节出现割裂感（如前文是古龙风，后文变成了网文风）。
* **元数据（Metadata）增强检索：** 单纯的向量检索可能返回不相关的结果。系统在存储向量时，会附加丰富的元数据标签：
  + {"chapter\_id": 10, "characters":, "location": "Tavern", "sentiment": "negative", "plot\_thread": "main"}  
    这一策略允许执行混合搜索（Hybrid Search）：先通过元数据过滤出“所有发生在酒馆且包含Alice的场景”，再在这些场景中进行向量相似度匹配。这种两阶段检索显著提升了上下文提取的精准度，确保模型生成的每一个字都建立在正确的情境之上5。

### 3.3 图谱与向量的协同：GraphRAG机制

本系统并未割裂使用图谱与向量，而是采用了**GraphRAG**的融合策略。在生成新内容时，系统执行以下步骤2：

1. **实体识别与链接：** 分析当前的写作意图，识别出关键实体（如“主角”、“断钢剑”）。
2. **子图提取（Subgraph Retrieval）：** 在Neo4j中查询这些实体的2跳（2-hop）邻域子图，获取所有相关的直接与间接关系（事实上下文）。
3. **向量召回（Vector Retrieval）：** 在ChromaDB中检索与当前情境语义相关的历史片段（风格与氛围上下文）。
4. **上下文拼接（Context Assembly）：** 将结构化的事实数据与非结构化的文本片段共同注入LLM的上下文窗口。图谱提供了骨架，向量提供了血肉，使得AI生成的文字既逻辑自洽又情感丰满。

## 4. 技术实施策略：基础设施与容器化部署

实现这一复杂架构需要坚实的底层基础设施。考虑到“完全本地运行”的核心需求，**Docker**与**Docker Compose**成为了编排Neo4j、ChromaDB及Python应用的最佳工具，确保了环境的可复现性与隔离性3。

### 4.1 容器化架构设计

docker-compose.yml文件是整个系统的部署蓝图，定义了各服务间的网络拓扑与存储挂载。

#### 4.1.1 Neo4j服务配置详解

Neo4j作为核心组件，其配置直接关系到知识图谱的可用性与扩展性。

* **镜像选择：** 推荐使用neo4j:latest或锁定稳定版本（如5.x），确保API兼容性。
* **端口映射：**
  + 7474:7474：HTTP端口，用于访问Neo4j Browser可视化界面，方便作者直观查看人物关系网。
  + 7687:7687：Bolt协议端口，提供给Python驱动（Driver）进行高性能数据交互3。
* **环境变量（Environment Variables）：**
  + NEO4J\_AUTH=neo4j/password：设置初始认证信息。生产环境中建议通过Docker Secrets或.env文件管理，避免明文泄露3。
  + NEO4J\_apoc\_export\_file\_enabled=true / NEO4J\_apoc\_import\_file\_enabled=true：启用APOC（Awesome Procedures on Cypher）插件库。APOC提供了大量高级图算法与数据转换工具，对于处理复杂的NLP任务至关重要13。
  + NEO4J\_PLUGINS=["apoc", "graph-data-science"]：自动安装图数据科学库，为未来可能引入的人物关系聚类分析预留能力13。
* **数据持久化（Volumes）：**
  + ./neo4j/data:/data：将容器内的数据库文件映射至宿主机目录。这是防止容器重启导致数据丢失的关键配置3。

#### 4.1.2 ChromaDB服务配置详解

ChromaDB采用\*\*客户端-服务器（Client-Server）\*\*模式部署，允许多个Python智能体进程并发访问同一向量存储。

* **镜像：** chromadb/chroma:latest。
* **端口：** 8000:8000，标准API接口4。
* **持久化关键配置：**
  + IS\_PERSISTENT=TRUE：必须显式设置为TRUE，否则Chroma默认运行在内存模式，重启即丢数据4。
  + PERSIST\_DIRECTORY=/chroma/chroma：指定容器内的存储路径。
  + ANONYMIZED\_TELEMETRY=FALSE：关闭匿名遥测，进一步强化隐私保护4。
* **卷挂载：**
  + ./chroma\_data:/chroma/chroma：将向量数据持久化到本地文件系统4。注意处理好文件权限问题，特别是在Linux环境下。

#### 4.1.3 网络编排与通信

所有服务置于同一Docker网络（如bridge模式）下，通过服务名（Service Name）互联，无需暴露不必要的端口到公网。

| **服务名称** | **内部端口** | **宿主机端口** | **网络别名** | **用途** |
| --- | --- | --- | --- | --- |
| neo4j | 7474, 7687 | 7474, 7687 | neo4j | 知识图谱后端 |
| chromadb | 8000 | 8000 | chromadb | 向量存储后端 |
| writer-app | N/A | N/A | app | 智能体逻辑代码 |

*注：开发阶段，Python脚本（writer-app）通常直接在宿主机（VSCode终端）运行，通过localhost:7687和localhost:8000访问数据库。生产部署时可将其容器化，通过内部别名neo4j和chromadb通信。*

### 4.2 网络安全与访问控制

对于“本地化”系统，安全重点在于防止意外的外部访问，特别是在使用宝塔面板（Baota Panel）或云服务器部署时。

* **端口绑定策略：** 在docker-compose.yml中，建议使用127.0.0.1:7474:7474的写法。这将端口强制绑定到回环接口，仅允许本机访问，物理隔绝了来自局域网或公网的连接尝试17。
* **防火墙穿透问题（UFW vs Docker）：** 在Ubuntu等Linux系统中，Docker默认会直接操作iptables，导致UFW防火墙规则失效（即ufw deny 8000无法阻止外部访问Docker映射的8000端口）。
  + **解决方案：** 必须使用ufw-docker工具或手动修改daemon.json禁用Docker的iptables管理功能（不推荐，易导致网络问题），最稳妥的方式仍是上述的“绑定到127.0.0.1”策略17。
* **宝塔面板集成：** 对于习惯GUI管理的用户，若使用宝塔面板部署Docker，需注意面板的安全组设置。若仅本地使用，切勿在面板中“放行”数据库端口。若需远程协作，建议通过SSH隧道（SSH Tunneling）而非直接暴露端口20。

## 5. 实施工作流：开发闭环与自动化集成

### 5.1 项目结构与模块化设计

为了支撑复杂的逻辑，项目文件结构应清晰分离数据、逻辑与配置1。

AI\_Novel\_Writer/

├── docker-compose.yml # 基础设施编排

├── requirements.txt # Python依赖 (neo4j, chromadb, openai/codex)

├── main.py # 主控调度脚本 (Orchestrator)

├── agents/ # 认知智能体模块

│ ├── planner.py # 构思规划

│ ├── writer.py # 章节撰写

│ ├── compressor.py # 信息抽取与压缩

│ ├── evaluator.py # 逻辑评价

│ └── editor.py # 风格润色

├── memory/ # 本地中间态存储 (JSON/MD)

│ ├── world.json

│ ├── characters.json

│ └── plot\_timeline.json

└── content/ # 小说输出目录

├── chapter\_001.md

└── chapter\_002.md

### 5.2 “大脑”与“记忆”的连接实现

#### 5.2.1 Python与Neo4j的交互逻辑

Python通过neo4j官方驱动与图数据库交互。

* **连接管理：** 使用GraphDatabase.driver(uri, auth)建立连接。在系统启动时，必须调用verify\_connectivity()方法进行健康检查，确保数据库就绪9。
* **数据写入模式：** 写作智能体并非直接写入。数据流是由“压缩智能体”解析文本后，生成三元组（Subject, Predicate, Object），再转化为Cypher的MERGE语句执行。MERGE指令能确保节点和关系存在时不重复创建，不存在时则新建，是维护图谱幂等性的关键9。
  + *代码示例逻辑：* MERGE (p:Character {name: "Hero"}) MERGE (l:Location {name: "Dungeon"}) MERGE (p)-->(l)。

#### 5.2.2 Python与ChromaDB的交互逻辑

* **客户端初始化：** 使用chromadb.HttpClient(host='localhost', port=8000)连接至Docker容器。注意这里需区分PersistentClient（用于直接读写本地文件）和HttpClient（用于连接服务模式）。在Docker部署下，HttpClient是标准选择14。
* **集合（Collection）隔离：** 建议创建不同的Collection来隔离数据类型，如narrative\_history（正文历史）、style\_snippets（风格片段）、character\_biographies（人物小传）。这有助于提高检索的相关性11。
* **数据持久性验证：** 在开发初期，务必测试重启容器后数据是否依然存在。常见错误是未正确挂载Volume，导致重启后client.list\_collections()返回空24。

### 5.3 剪贴板自动化：人机交互的最后一步

为了让作者在VSCode中获得无缝体验，系统引入了基于剪贴板监听的自动化脚本。

* **机制：** 利用Python的pyperclip库编写后台脚本，实时监控系统剪贴板的变化25。
* **场景：** 当作者在VSCode中选中一段AI生成的文字并复制时，后台脚本检测到新内容，自动触发“评价智能体”进行分析，并将分析结果（如“检测到被动语态过多”）通过系统通知或弹窗反馈给作者。或者，作者复制一段大纲，脚本自动识别并调用“Writer Agent”开始生成正文，并将结果自动粘贴回编辑器或存入文件。这种“Copy-to-Compute”模式极大地减少了在编辑器与命令行之间切换的摩擦成本27。

## 6. 深度洞察与未来演进

### 6.1 “幻觉”的辩证法：从Bug到Feature

在金融或医疗领域的RAG系统中，幻觉（Hallucination）是致命缺陷。但在小说创作中，适度的幻觉即是**创造力**。本架构的核心优势在于它能够平衡“约束”与“发散”。Neo4j图谱作为不可撼动的“世界观宪法”，提供了硬性约束；而在图谱未定义的空白区域，Writer智能体被允许使用较高的Temperature参数进行发散性联想。系统通过这种方式将“有界创造力”工程化，既避免了胡编乱造，又保留了AI惊艳的脑洞。

### 6.2 作者角色的范式转移

本系统将作者从“逐字生成者”重塑为“叙事管理者”。作者不再需要苦思每一个形容词，而是专注于定义图谱中的约束条件（如“设定这个国家禁止魔法”）和规划大纲目标。Editor智能体的存在，使得作者的介入点后移至质量控制阶段。这种转变类似于从汇编语言编程进化到高级语言编程——作者编写逻辑（Plot），AI负责编译（Writing），极大地释放了人类在宏观叙事上的构思能力。

### 6.3 云原生扩展路径

尽管当前设计聚焦于本地化，但其容器化架构天然具备\*\*云原生（Cloud-Native）\*\*基因。随着小说篇幅增长至数百万字，本地向量检索可能出现延迟。此时，无需重构核心逻辑，只需修改docker-compose配置，即可将ChromaDB容器平滑迁移至分布式的云端向量数据库集群，或将Neo4j切换至Aura云服务。这种弹性架构保证了系统能够伴随创作者的野心无限扩展3。

### 6.4 活着的进度条：状态机叙事

“情节进度线”模块不应仅被视为静态记录，它本质上是一个**状态机（State Machine）**。通过在JSON中追踪“未解决的伏笔（Unresolved Foreshadowing）”队列，系统维护了一组“开放循环（Open Loops）”。规划智能体在生成新章节大纲时，会被强制检查这个队列，确保切科夫之枪（Chekhov's Gun）在后续章节中必然打响。这种机制赋予了AI类似资深作家的统筹能力，使得故事结构严丝合缝，而非松散的流水账。

## 结论

本报告所述的“本地化AI小说写作系统”不仅仅是一个工具的堆砌，而是对\*\*代理式AI（Agentic AI）\*\*在创意领域应用的一次深度实践。通过将大语言模型的概率生成能力锚定在知识图谱的确定性事实与向量数据库的语义深度之上，该框架成功解决了一致性与记忆力的行业痛点。Docker的使用确保了技术栈的普适性与数据主权，而与VSCode的深度集成则将写作提升到了软件工程的严谨高度。这不仅是现代小说家的辅助工具，更是通向未来“人机共创”时代的认知义肢。

## 数据概览表：系统核心组件对比

| **组件名称** | **技术选型** | **核心职责** | **数据类型** | **记忆性质** |
| --- | --- | --- | --- | --- |
| **Concept Planner** | Codex (Local) | 叙事结构规划、大纲生成 | 文本/JSON | 认知处理 (CPU) |
| **Writer Agent** | Codex (Local) | 正文文本生成、多版本草稿 | 文本 | 瞬时工作记忆 (RAM) |
| **Compressor** | NLP/LLM | 信息抽取、摘要生成 | 结构化数据 | 记忆转化 (RAM->HDD) |
| **World Graph** | **Neo4j** | 存储世界观、人物关系、因果链 | 节点与边 (Graph) | **符号记忆 (Symbolic)** - 事实与逻辑 |
| **Narrative Store** | **ChromaDB** | 存储正文片段、风格样本 | 向量 (Vectors) | **语义记忆 (Semantic)** - 氛围与语感 |
| **Infrastructure** | Docker Compose | 服务编排、环境隔离 | 容器 (Containers) | 运行环境 (OS) |

**参考资料来源标识：**

* 系统框架设计与智能体定义：1
* Neo4j图数据库架构与配置：2
* ChromaDB向量库与持久化：4
* Docker容器化与网络安全：3
* Python集成与自动化脚本：22
* 宝塔面板与内容过滤：20

#### Works cited

1. 本地化 AI 小说写作系统框架设计.pdf
2. GraphRAG Explained: Building Knowledge-Grounded LLM Systems with Neo4j and LangChain | by DhanushKumar | Dec, 2025 | Towards AI, accessed December 23, 2025, <https://pub.towardsai.net/graphrag-explained-building-knowledge-grounded-llm-systems-with-neo4j-and-langchain-017a1820763e>
3. Deploy a Neo4j standalone server using Docker Compose - Operations Manual, accessed December 23, 2025, <https://neo4j.com/docs/operations-manual/current/docker/docker-compose-standalone/>
4. Running Chroma, accessed December 23, 2025, <https://cookbook.chromadb.dev/running/running-chroma/>
5. ChromaDB: The Long-Term Semantic Memory Engine Behind My Multi-Agent System | by Sendoa Moronta | Nov, 2025 | Medium, accessed December 23, 2025, <https://medium.com/@sendoamoronta/chromadb-the-long-term-semantic-memory-engine-behind-my-multi-agent-system-4261fe0610ce>
6. Knowledge Graph Extraction and Challenges - Graph Database & Analytics - Neo4j, accessed December 23, 2025, <https://neo4j.com/blog/developer/knowledge-graph-extraction-challenges/>
7. How to Build a Knowledge Graph in 7 Steps - Neo4j, accessed December 23, 2025, <https://neo4j.com/blog/knowledge-graph/how-to-build-knowledge-graph/>
8. Knowledge Graph - Graph Database & Analytics - Neo4j, accessed December 23, 2025, <https://neo4j.com/use-cases/knowledge-graph/>
9. How to Connect Python to Neo4j Graph Database: Complete Guide - Mito, accessed December 23, 2025, <https://www.trymito.io/blog/how-to-connect-python-to-neo4j-graph-database-complete-guide>
10. How to Convert Unstructured Text to Knowledge Graphs Using LLMs - Neo4j, accessed December 23, 2025, <https://neo4j.com/blog/developer/unstructured-text-to-knowledge-graph/>
11. Guide to Chroma DB: A Vector Store for Your Generative AI LLMs - Analytics Vidhya, accessed December 23, 2025, <https://www.analyticsvidhya.com/blog/2023/07/guide-to-chroma-db-a-vector-store-for-your-generative-ai-llms/>
12. Build and host a ChromaDB vector database - DataRobot docs, accessed December 23, 2025, <https://docs.datarobot.com/en/docs/gen-ai/genai-code/chromadb-vdb.html>
13. Simple Graph Database Setup with Neo4j and Docker Compose | by Matthew Ghannoum, accessed December 23, 2025, <https://medium.com/@matthewghannoum/simple-graph-database-setup-with-neo4j-and-docker-compose-061253593b5a>
14. Docker - Chroma Docs, accessed December 23, 2025, <https://docs.trychroma.com/guides/deploy/docker>
15. Configuration - Chroma Cookbook, accessed December 23, 2025, <https://cookbook.chromadb.dev/core/configuration/>
16. How to deploy chroma database (vector database) in production - Stack Overflow, accessed December 23, 2025, <https://stackoverflow.com/questions/76505538/how-to-deploy-chroma-database-vector-database-in-production>
17. chaifeng/ufw-docker: To fix the Docker and UFW security flaw without disabling iptables - GitHub, accessed December 23, 2025, <https://github.com/chaifeng/ufw-docker>
18. What is the best practice of docker + ufw under Ubuntu [closed] - Stack Overflow, accessed December 23, 2025, <https://stackoverflow.com/questions/30383845/what-is-the-best-practice-of-docker-ufw-under-ubuntu>
19. TIL: Docker overrides ufw and iptables rules by injecting it's own rules : r/selfhosted - Reddit, accessed December 23, 2025, <https://www.reddit.com/r/selfhosted/comments/1atjsra/til_docker_overrides_ufw_and_iptables_rules_by/>
20. Baota Panel + Rancher + Alibaba Cloud Image Repository + Docker + Kubernetes: Adding Clusters and Deploying Web Applications - 痴者工良的博客, accessed December 23, 2025, <https://www.whuanle.cn/archives/22211>
21. BT Panel (宝塔面板) - AingDesk Docs, accessed December 23, 2025, <https://docs.aingdesk.com/Installation/btpanel/>
22. Connection - Neo4j Python Driver Manual, accessed December 23, 2025, <https://neo4j.com/docs/python-manual/current/connect/>
23. chroma in the docker cannot be connected from another docker service - Stack Overflow, accessed December 23, 2025, <https://stackoverflow.com/questions/79171453/chroma-in-the-docker-cannot-be-connected-from-another-docker-service>
24. [Bug]: Docker config.yaml breaks persistence · Issue #4330 · chroma-core/chroma - GitHub, accessed December 23, 2025, <https://github.com/chroma-core/chroma/issues/4330>
25. Copy and Paste Text to the Clipboard with pyperclip in Python | note.nkmk.me, accessed December 23, 2025, <https://note.nkmk.me/en/python-pyperclip-usage/>
26. Detect clipboard copy/paste event and modify clipboard contents - Ask Ubuntu, accessed December 23, 2025, <https://askubuntu.com/questions/1167026/detect-clipboard-copy-paste-event-and-modify-clipboard-contents>
27. TakamiChie/ClipboardMonitor: Clipboard monitor tool. It displays the text data stored in the clipboard. - GitHub, accessed December 23, 2025, <https://github.com/TakamiChie/ClipboardMonitor>
28. Neo4j Tutorial: Using And Querying Graph Databases in Python - DataCamp, accessed December 23, 2025, <https://www.datacamp.com/tutorial/neo4j-tutorial>
29. Docker-specific configuration settings - Operations Manual - Neo4j, accessed December 23, 2025, <https://neo4j.com/docs/operations-manual/current/docker/ref-settings/>
30. Publishing and exposing ports | Docker Docs, accessed December 23, 2025, <https://docs.docker.com/get-started/docker-concepts/running-containers/publishing-ports/>
31. How do I copy a string to the clipboard? - python - Stack Overflow, accessed December 23, 2025, <https://stackoverflow.com/questions/579687/how-do-i-copy-a-string-to-the-clipboard>