

Complete Guide: Setting Up a RAG Agent with Flowise Document Store

Overview

This comprehensive tutorial will guide you through setting up a **Retrieval-Augmented Generation (RAG) Agent** using Flowise with a document store. You'll learn how to:

- Create a new Document Store in Flowise
- Configure web scraping with Cheerio
- Set up Pinecone vector database
- Configure embeddings and record management
- Implement the complete RAG pipeline

Prerequisites

- Flowise installed and running (see flowiseSetupGuide.md)
- Pinecone account (free tier available)
- Basic understanding of RAG concepts
- Target website or documents for scraping

What is RAG (Retrieval-Augmented Generation)?

RAG combines the power of large language models with external knowledge retrieval. Instead of relying solely on pre-trained knowledge, RAG systems:

1. **Retrieve** relevant information from external sources
2. **Augment** the AI's context with this retrieved information
3. **Generate** more accurate and up-to-date responses

Step-by-Step Tutorial

Step 1: Create a New Document Store in Flowise

Document Store

Store and upsert documents for LLM retrieval (RAG)

The screenshot shows a modal dialog box titled "Add New Document Store". Inside the dialog, there is a "Name" field containing "MScproject" and a larger "Description" field which is currently empty. At the bottom right of the dialog are two buttons: "Cancel" and a blue "Add" button.

1. Access Flowise Interface

- Open your Flowise installation in a web browser
- Navigate to the main dashboard

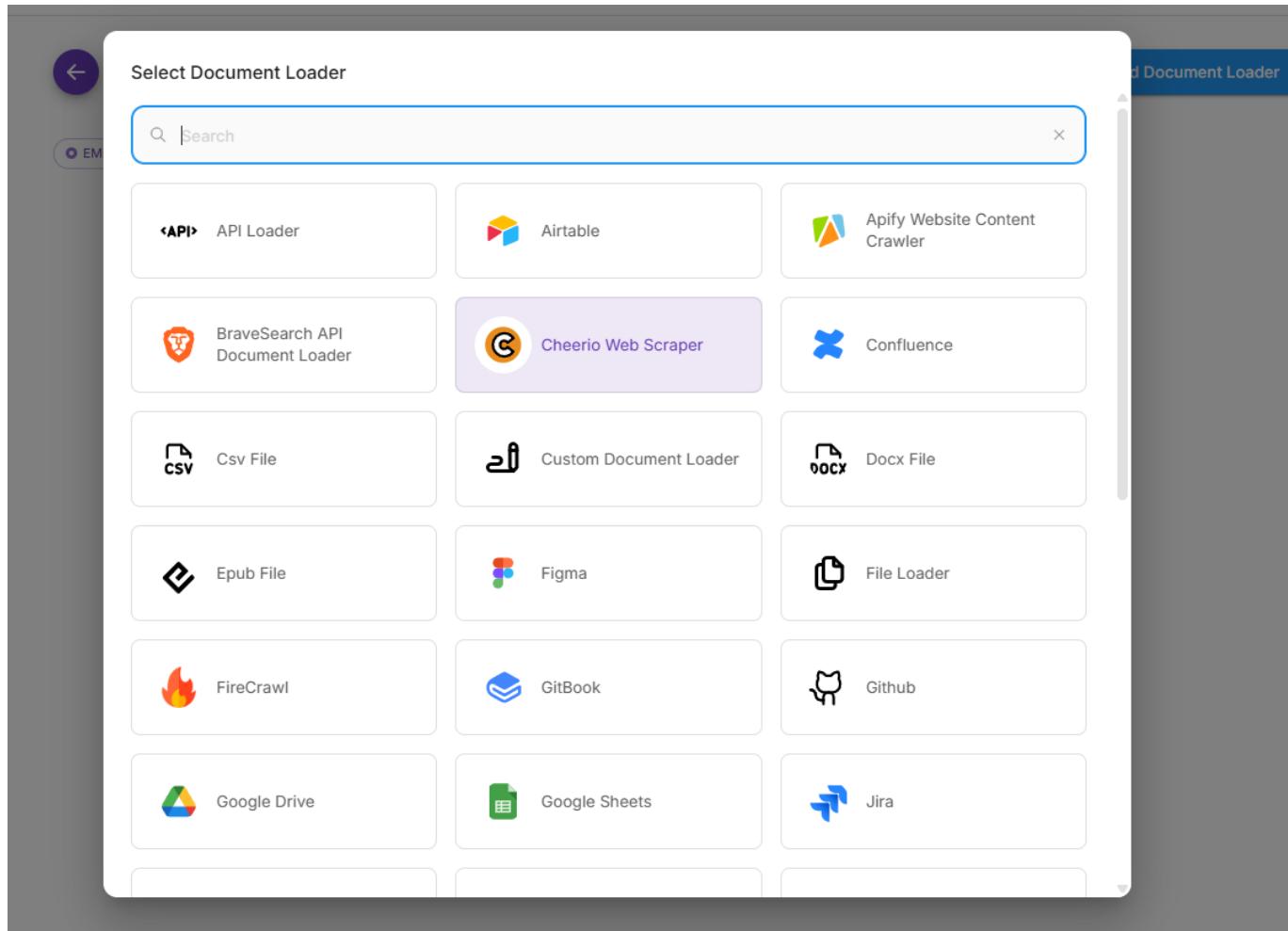
2. Create Document Store

- Look for the "Document Store" section
- Click on "**+ New Document Store**" or similar option
- This will open the document store configuration interface

Key Points:

- Document stores are the foundation of RAG systems
- They manage how documents are processed and stored
- Essential for creating knowledge bases from various sources

Step 2: Configure Cheerio Web Scraper



1. Select Web Scraping Method

- Choose "**Cheerio Web Scraper**" as your data source
- Cheerio is a server-side jQuery implementation perfect for web scraping

2. Configure Scraper Settings

- Set up the scraper parameters
- Configure selectors for content extraction
- Define how the scraper should handle different HTML elements

Why Cheerio?

- Lightweight and fast
- Excellent for structured content extraction
- Works well with static websites
- Supports CSS selectors for precise content targeting

Step 3: Set Up Web Crawling Parameters

The screenshot shows the Cheerio Web Scraper configuration page. On the left, there are several input fields and dropdowns:

- Cheerio Web Scraper Loader Name:** fiction
- URL ***: https://enoche-sit.github.io/Blog/docs/12CCNAStory/storyline
- Get Relative Links Method:** Web Crawl
- Get Relative Links Limit:** 10 (with a note: "Retrieving all links might take long time, and all links will be upserted again if the flow's state changed (eg: different URL, chunk size, etc.)")
- Selector (CSS):** article p
- Additional Metadata:** An empty object {} with 0 items.
- Omit Metadata Keys:** key1, key2, key3.nestedKey1

On the right, the results are displayed in a grid of 14 chunks:

- #1. Characters: 999**: 網道劍影錄序章：風雨前夕（原文照錄：交代故事背景，引出互聯山莊、黑風堡、網道真解以及最初的江湖形勢。）卷一：初窮門徑 - 基礎心法第一章：萬物有靈 - 識器（Day 1: Networking Devices - 網絡設備）葉辰，互聯山莊一名尋常弟子，得隱世長老風明暗中指點，開始了他的修行之路。風明首先向他介紹了網絡中的“靈物”：「Router導引訣」、「Switch分流指」、「Firewall御魔心經」、「Access
- #2. Characters: 999**: Switching - Part 1 - 以太網局域網交換基礎) 風明開始傳授葉辰局域陣法之技，核心是「MAC learning and aging見聞不忘」（MAC地址學習與老化——見聞不忘）。葉辰在一組互聯的訓練僞偽（設備）上練習，學習交換機如何建立對周圍“人物”（設備）位置的認知。第六章：動靜流轉 - 變陣 (Day 6: Ethernet LAN Switching - Part 2 - 以太網局域網交換進階) 葉辰深入鑽研交換之術，包括
- #3. Characters: 933**: - 問道 (Day 11: Routing Fundamentals - Part 1 & Static Routing - Part 2 - 路由基礎與靜態路由) (合併“路由基礎 - Part 1”與“靜態路由 - Part 2”，因其在原故事中緊密相連) 風明揭示了在局域之外尋找路徑的藝術：「Routing table尋幽探勝圖」（路由表——尋幽探勝圖）。葉辰學習解讀這份“地圖”，理解「Prefix前路漫漫」、「Network mask遮天蔽日」和「Next hop柳暗花明」。隨後他練習「Static
- #4. Characters: 981**: 15: Subnetting (VLSM) - Part 3 - 可變長子網掩碼) 風明引入了可變長子網掩碼 (VLSM)，一種“量體裁衣”般劃分網絡大小的技藝。葉辰學會了優化地址分配，這對於山莊“真氣”通路的有效管理至關重要。卷二：江湖險阻 - 歷練之路（黑風堡的陰影日益逼近，葉辰的武功面臨考驗。）第十六章：虛實分界 - 內域 (Day 16: VLANs - Part 1 - VLAN基礎) 隨著黑風堡的傳聞漸緊，風明傳授葉辰「VLANs
- #5. Characters: 995**: Part 1 - 生成樹協議基礎) 為防止因冗餘路徑導致的災難性“真氣風暴”（廣播風暴），葉辰學習了生成樹協議 (STP) 的基礎。
- #6. Characters: 984**: - 初會 (Day 26: OSPF - Part 1 - OSPF協議基礎) 修煉重點轉向一門強大且廣受推崇的路由技藝：「OSPFv2群龍聚首訣」。

1. Configure Crawling Settings

- **URL:** Enter the target website URL
- **Crawl Depth:** Set how many levels deep to crawl
- **Content Selectors:** Define which elements to extract
- **Exclusion Rules:** Set patterns to avoid (ads, navigation, etc.)

2. Advanced Options

- **Delay:** Set crawl delays to be respectful
- **Headers:** Configure user agent and other headers
- **Rate Limiting:** Prevent overwhelming the target server

Best Practices:

- Always respect robots.txt
- Use reasonable crawl delays
- Focus on content-rich pages
- Exclude navigation and advertising content

Step 4: Create Pinecone Vector Database Index

Welcome to your project

You're on the path to building remarkably better applications.
Create an index to get started.

[Create index](#)

[Load from dataset](#)

The screenshot shows the Pinecone API quickstart page. It includes sections for 'Install Pinecone' (with a code snippet for pip install pinecone), 'Initialize' (with a code snippet for initializing the client using an API key), and 'Create index' (with a note about creating an index integrated with an embedding model).

API quickstart

Python

Install Pinecone
Ready to get started with Pinecone? First, install the Python SDK:

```
1 pip install pinecone
```

Initialize
Next, use your API key to initialize your client:

```
1 from pinecone import Pinecone, ServerlessSpec
2
3 pc = Pinecone(api_key="*****-****-****-****-*****")
```

Create index
Then [create an index](#) that is integrated with an embedding model hosted by Pinecone.
With integrated models, you upsert and search with text and have Pinecone generate

1. Access Pinecone Console

- Log into your Pinecone account at [pinecone.io](#)
- Navigate to the indexes section

2. Create New Index

- Click "**Create Index**"
- Choose appropriate settings:
 - **Name:** Give your index a descriptive name
 - **Environment:** Select your preferred region
 - **Pod Type:** Choose based on your needs (p1.x1 for testing)

Index Configuration:

- **Dimension:** Will be set based on embedding model (typically 1536 for AWS Text embedding model)
- **Metric:** Usually "cosine" for text similarity
- **Pods:** Start with 1 pod for development

Step 5: Configure Embedding Dimensions

Create a new index

The screenshot shows the 'Create a new index' form. At the top, there's a 'Default / fiction' dropdown. Below it, the 'Configuration' section includes 'Vector type' set to 'Dense', 'Dimension' set to '1,536', and 'Metric' set to 'cosine'. A 'Custom settings' checkbox is checked. In the 'Capacity mode' section, 'Serverless' is selected, with a note that charges are based on data storage, reads, and writes. At the bottom, it says you're on the 'Starter plan' and has 'Cancel' and 'Create index' buttons.

Default / fiction

Configuration

Vector type ⓘ
Dense

Dimension ⓘ
1,536

Metric ⓘ
cosine

Custom settings

Capacity mode

Serverless
Charges based on data storage, reads, and writes.

You are currently on the
Starter plan ⓘ

Cancel Create index

1. Set Embedding Dimensions

- Configure the embedding dimension to **1536**
- This matches OpenAI's text-embedding-ada-002 model
- Ensure consistency between your embedding model and vector database

2. Embedding Model Selection

- Choose your preferred embedding model
- OpenAI Ada-002 is popular for its quality and speed
- Ensure the dimensions match your Pinecone index

Common Embedding Dimensions:

- OpenAI Ada-002:** 1536 dimensions
- Sentence Transformers:** 384-768 dimensions
- Cohere:** 4096 dimensions

Step 6: Generate Pinecone API Key

The screenshot shows the Pinecone API keys management interface. On the left, there's a sidebar with links like 'Get started', 'Database', 'Assistant', 'Inference', 'API keys' (which is selected and highlighted in blue), and 'Manage'. Below that is a 'STARTER USAGE' section showing RUs, WUs, and Storage usage. The main area is titled 'API keys' and lists a single key named 'default' created on '2025/9/16'. A modal window titled 'Generate new API key' is open, prompting for an 'API key name' (set to 'mscproject') and 'Permissions' (set to 'All'). There's also a note about upgrading for advanced permissions. At the bottom of the modal are 'Cancel' and 'Create key' buttons.

1. Navigate to API Keys

- In Pinecone console, go to "**API Keys**" section
- Click "**Create API Key**"

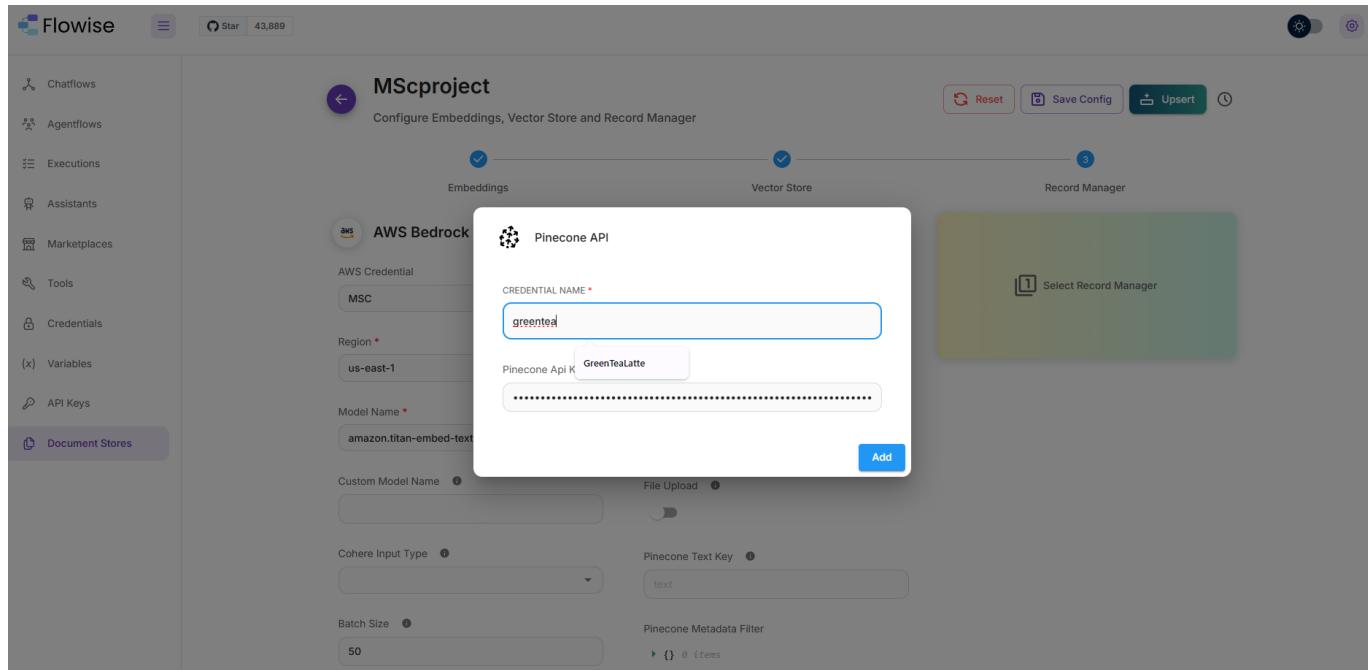
2. Configure API Key

- **Name:** Give it a descriptive name (e.g., "flowise-rag-agent")
- **Environment:** Select the same environment as your index
- **Permissions:** Ensure it has read/write access to your index

3. Save the Key

- **⚠️ Important:** Copy and save the API key immediately
- Store it securely - you won't be able to see it again
- Consider using environment variables for security

Step 7: Add API Key to Flowise



1. Navigate to Pinecone Configuration

- Return to your Flowise document store setup
- Find the Pinecone vector store configuration

2. Enter API Credentials

- **API Key:** Paste your Pinecone API key
- **Environment:** Enter your Pinecone environment (e.g., "us-east1-gcp")
- **Index Name:** Enter the name of your created index

Security Best Practices:

- Never commit API keys to version control
- Use environment variables in production
- Regularly rotate API keys
- Monitor API usage

Step 8: Configure Index Settings

The screenshot shows the MScproject configuration interface. At the top, there are three tabs: 'Embeddings' (green checkmark), 'Vector Store' (green checkmark), and 'Record Manager' (yellow background with a red '3'). Below the tabs, there are two main sections: 'AWS Bedrock Embeddings' and 'Pinecone'. The 'AWS Bedrock Embeddings' section includes fields for AWS Credential (MSC), Region (us-east-1), Model Name (amazon.titan-embed-text-v1), Custom Model Name, Cohere Input Type, and Batch Size (50). The 'Pinecone' section includes fields for Connect Credential (greentea), Pinecone Index (fiction), Pinecone Namespace (my-first-namespace), File Upload, Pinecone Text Key (text), and Pinecone Metadata Filter. A large yellow callout box labeled 'Select Record Manager' is positioned over the Record Manager tab.

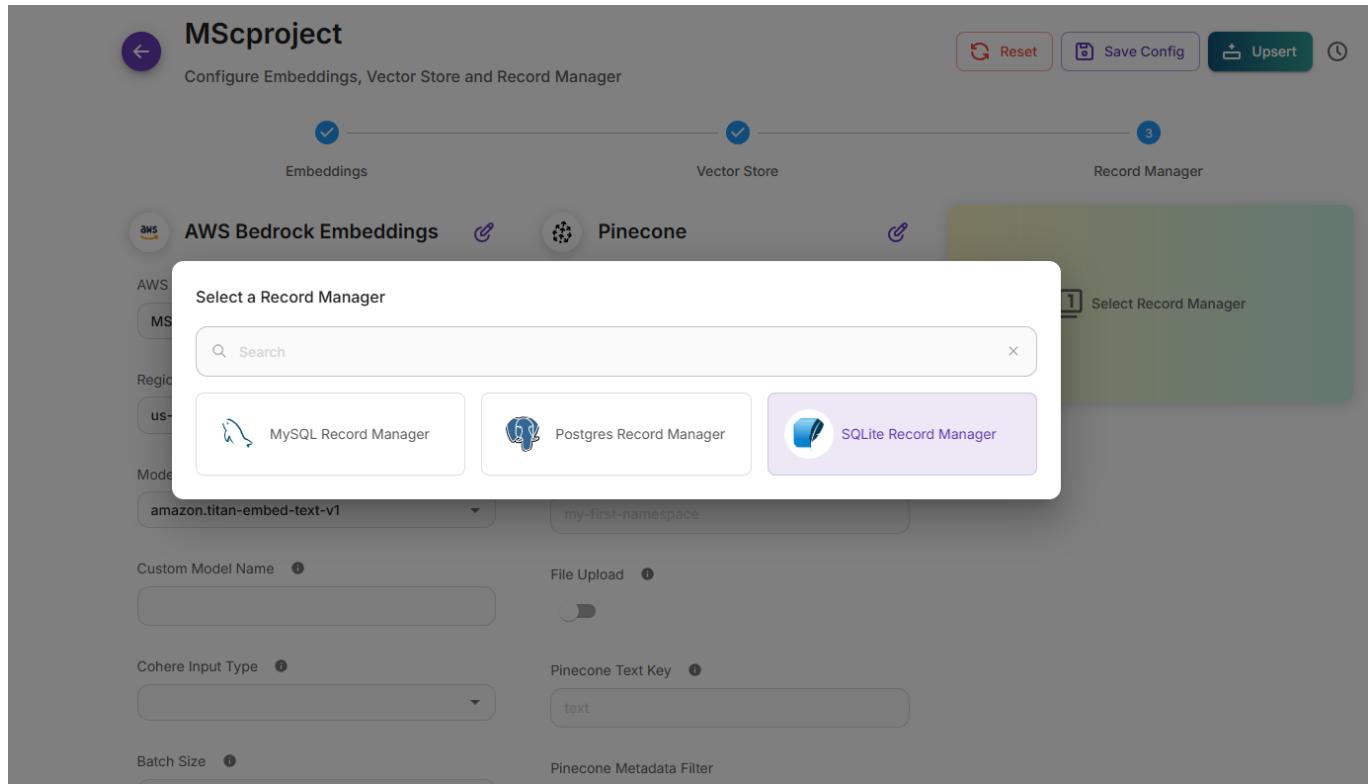
1. Complete Index Configuration

- Verify all Pinecone settings are correct
- **Index Name:** Confirm it matches your Pinecone index
- **Namespace:** Optional, useful for organizing data
- **Top K:** Number of similar documents to retrieve (typically 3-10)

2. Vector Store Settings

- **Metadata:** Configure what metadata to store
- **Text Key:** Field name for document content
- **Filter:** Optional filters for specific document types

Step 9: Set Up SQLite Record Manager



1. Choose Record Manager

- Select "**SQLite Record Manager**"
- This tracks which documents have been processed
- Prevents duplicate processing and enables incremental updates

2. Record Manager Benefits

- **Deduplication:** Prevents processing same content multiple times
- **Incremental Updates:** Only processes new/changed content
- **Cleanup:** Helps remove outdated content
- **Tracking:** Maintains history of processed documents

Why SQLite?

- Lightweight and self-contained
- No additional database setup required
- Perfect for development and small-scale deployments
- Easy to backup and migrate

Step 10: Configure Full Cleanup Mode

The screenshot shows the configuration interface for a Vector Store and Record Manager. At the top, there are three main sections: Vector Store, Record Manager, and a central configuration area.

Pinecone Configuration:

- Connect Credential ***: Set to "greentea".
- Pinecone Index ***: Set to "fiction".
- Pinecone Namespace**: Set to "my-first-namespace".
- File Upload**: A toggle switch is off.
- Pinecone Text Key**: Set to "text".
- Pinecone Metadata Filter**: An empty input field.

SQLite Record Manager Configuration:

- Additional Connection Configuration**: A dropdown menu is open, showing three options:
 - None**: No clean up of old content.
 - Incremental**: Delete previous versions of the content if content of the source document has changed. Important!! Sourceld Key must be specified and document metadata must contains the specified key.
 - Full**: Same as incremental, but if the source document has been deleted, it will be deleted from vector store as well, incremental mode will not.
- Sourceld Key**: Set to "source".

1. Select Cleanup Strategy

- Choose "**Full Cleanup**" mode
- This ensures old, outdated documents are removed
- Maintains fresh, relevant content in your vector store

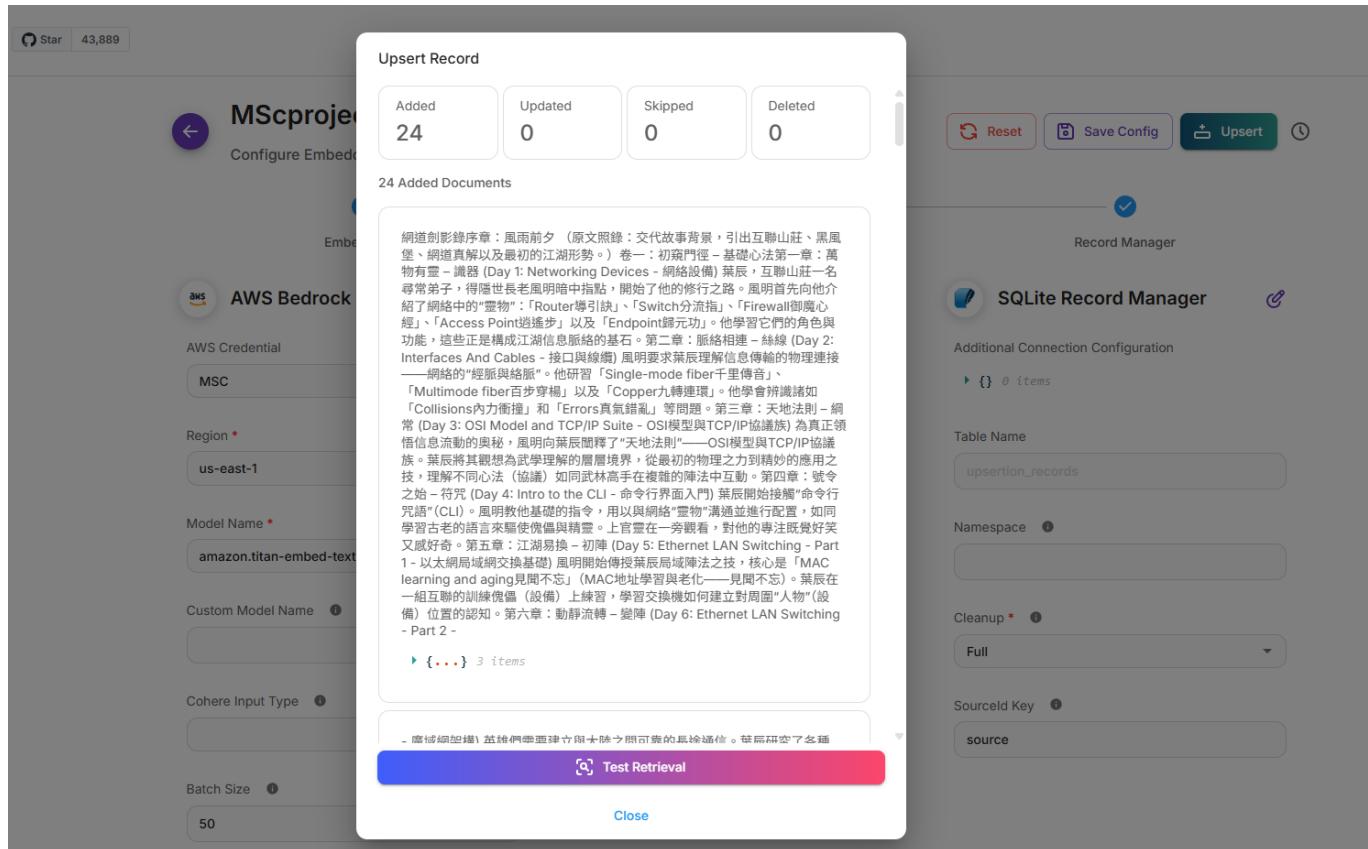
2. Cleanup Options

- **Full**: Removes all records not in current batch
- **Incremental**: Only removes explicitly marked records
- **None**: Never removes records (accumulates over time)

When to Use Full Cleanup:

- Content changes frequently
- Want to prevent stale information
- Need complete content refresh
- Have very large, stable datasets
- Want to preserve historical versions

Step 11: Execute the Upsert Operation



1. Start the Upsert Process

- Click "**Upsert**" to begin processing
- This will:
 - Scrape content from your target website
 - Convert text to embeddings
 - Store vectors in Pinecone
 - Update the record manager

2. Monitor Progress

- Watch for processing status updates
- Check logs for any errors or warnings
- Verify documents are being processed correctly

What Happens During Upsert:

1. **Scraping:** Cheerio extracts content from web pages
2. **Chunking:** Large documents are split into smaller segments
3. **Embedding:** Text is converted to numerical vectors
4. **Storage:** Vectors are uploaded to Pinecone
5. **Recording:** Process is logged in SQLite record manager

Testing Your RAG Agent

Verify Document Store Setup

```
# Check Pinecone index stats  
# You can do this through Pinecone console or API  
  
# Verify SQLite record manager  
# Check the created database file for processed records
```

Create a Chat Flow

1. Create New Chatflow

- Use your document store as knowledge base
- Connect to your preferred LLM (Nova etc.)
- Configure retrieval parameters

2. Test Queries

- Ask questions related to your scraped content
- Verify accurate retrieval and relevant responses
- Check citation and source attribution

Troubleshooting Common Issues

Connection Problems

Issue: Cannot connect to Pinecone **Solution:**

- Verify API key is correct
- Check environment and region settings
- Ensure index name matches exactly

Scraping Issues

Issue: No content being scraped **Solution:**

- Verify target URL is accessible
- Check CSS selectors are correct
- Ensure robots.txt allows scraping
- Test with a simpler page first

Embedding Problems

Issue: Dimension mismatch error **Solution:**

- Ensure Pinecone index dimension matches embedding model
- Common dimensions: 1536 (OpenAI), 768 (many others)
- Recreate index with correct dimensions if needed

Performance Issues

Issue: Slow upsert process **Solution:**

- Reduce batch size
- Add delays between requests
- Consider upgrading Pinecone plan
- Optimize content selectors

Best Practices

Content Strategy

- **Quality over Quantity:** Focus on high-value, relevant content
- **Regular Updates:** Schedule periodic re-indexing
- **Content Curation:** Remove or filter low-quality content
- **Chunking Strategy:** Balance chunk size for context vs. precision

Security

- **API Key Management:** Use environment variables
- **Access Control:** Limit API key permissions
- **Rate Limiting:** Respect service limits
- **Monitoring:** Track usage and costs

Performance Optimization

- **Batch Processing:** Process documents in batches
- **Parallel Processing:** Use multiple workers when possible
- **Caching:** Cache frequently accessed embeddings
- **Index Optimization:** Regularly maintain and optimize indices

Advanced Configuration

Custom Embedding Models

```
# Example configuration for custom embeddings
embedding:
  model: "sentence-transformers/all-MiniLM-L6-v2"
  dimension: 384
  batch_size: 32
```

Advanced Chunking

```
# Sophisticated chunking strategy
chunking:
  strategy: "recursive"
  chunk_size: 1000
  chunk_overlap: 200
  separators: ["\n\n", "\n", " ", ""]
```

Metadata Enhancement

```
# Rich metadata configuration
metadata:
  include:
    - url
    - title
    - date_scraped
    - content_type
    - word_count
  custom_fields:
    - department
    - priority_level
```

Monitoring and Maintenance

Regular Health Checks

1. Vector Store Health

- Monitor index size and utilization
- Check query performance metrics
- Verify data freshness

2. Content Quality

- Review retrieved content accuracy
- Monitor user feedback
- Update content sources as needed

3. System Performance

- Track response times
- Monitor resource usage
- Optimize based on usage patterns

Conclusion

You've successfully created a complete RAG Agent with Flowise! This setup provides:

- **Automated Web Scraping** with Cheerio
- **Vector Storage** with Pinecone
- **Embedding Management** with OpenAI
- **Record Tracking** with SQLite
- **Incremental Updates** with cleanup management

Next Steps

1. **Expand Content Sources:** Add more websites or document types
2. **Optimize Retrieval:** Fine-tune similarity thresholds and result counts

3. **Enhance Metadata:** Add more contextual information
4. **Monitor Performance:** Set up analytics and monitoring
5. **Scale Up:** Consider production deployment strategies

Additional Resources

- [Flowise Documentation](#)
 - [Pinecone Best Practices](#)
 - [RAG System Design Patterns](#)
 - [Vector Database Optimization](#)
-

Happy Building! 

Remember: The key to a successful RAG system is high-quality, relevant content combined with proper retrieval configuration. Start simple and iterate based on user feedback and performance metrics.