GroundRAG - Complete Setup Guide

Step-by-Step Installation & Running Instructions

Project: GroundRAG - Multi-Source Agentic RAG System
Tech Stack: FastAPI + Streamlit + smolagents + Qdrant + Gemini 2.0 Flash
Date: October 23, 2025

Table of Contents

Prerequisites {#prerequisites}

Required Software

1. **Python 3.11+**
   - Download from: https://www.python.org/downloads/
   - Verify: `python --version`
2. **Docker & Docker Compose** (for Qdrant)
   - Download from: https://www.docker.com/get-started
   - Verify: `docker --version` and `docker-compose --version`
3. **Git** (optional, for version control)
   - Download from: https://git-scm.com/downloads
   - Verify: `git --version`

Required API Keys

1. **Google AI API Key** (REQUIRED)
   - Get it from: https://makersuite.google.com/app/apikey
   - OR from Google Cloud Console: https://console.cloud.google.com/
   - Enable Gemini API in your Google Cloud project

Project Structure Setup {#project-structure}

Step 1: Create Project Directory

Open terminal/command prompt and create the project structure:

```
# Create main project directory
mkdir groundrag
cd groundrag

# Create backend structure
mkdir -p backend/app/config
mkdir -p backend/app/api/v1
mkdir -p backend/app/agents/tools
mkdir -p backend/app/services
mkdir -p backend/app/models
mkdir -p backend/app/db
mkdir -p backend/app/utils
mkdir -p backend/tests

# Create frontend structure
mkdir -p frontend/components
mkdir -p frontend/services
mkdir -p frontend/utils
```

Your structure should look like this:

```
groundrag/
├── backend/
│   ├── app/
│   │   ├── __init__.py
│   │   ├── main.py
│   │   ├── config/
│   │   │   ├── __init__.py
│   │   │   └── settings.py
│   │   ├── api/
│   │   │   ├── __init__.py
│   │   │   └── v1/
│   │   │       ├── __init__.py
│   │   │       ├── upload.py
│   │   │       ├── chat.py
│   │   │       └── sources.py
│   │   ├── agents/
│   │   │   ├── __init__.py
```

```
│   │   │   ├── masa_agent.py
│   │   │   └── tools/
│   │   │       ├── __init__.py
│   │   │       ├── retriever_tool.py
│   │   │       ├── image_tool.py
│   │   │       ├── audio_tool.py
│   │   │       └── video_tool.py
│   │   ├── services/
│   │   │   ├── __init__.py
│   │   │   ├── ingestion_service.py
│   │   │   ├── chat_service.py
│   │   │   ├── source_service.py
│   │   │   └── vector_service.py
│   │   ├── models/
│   │   │   ├── __init__.py
│   │   │   ├── request_models.py
│   │   │   └── response_models.py
│   │   ├── db/
│   │   │   ├── __init__.py
│   │   │   └── qdrant_client.py
│   │   └── utils/
│   │       ├── __init__.py
│   │       └── chunking.py
│   ├── requirements.txt
│   ├── Dockerfile
│   └── .env
│
├── frontend/
│   ├── app.py
│   ├── components/
│   │   ├── __init__.py
│   │   ├── upload_panel.py
│   │   ├── chat_panel.py
│   │   └── citation_display.py
│   ├── services/
│   │   ├── __init__.py
│   │   └── api_client.py
│   ├── utils/
│   │   ├── __init__.py
│   │   └── session_manager.py
│   ├── requirements.txt
│   ├── Dockerfile
│   └── .env
│
├── docker-compose.yml
├── .gitignore
└── README.md
```

File Organization {#file-organization}

Step 2: Download and Place Files

I've generated **all code files** for you. Download them and place them in the correct locations:

Configuration Files (Root Directory)

1. **docker-compose.yml** → `groundrag/docker-compose.yml`
2. **.gitignore** → `groundrag/.gitignore`

Backend Files

1. **backend_requirements.txt** → `groundrag/backend/requirements.txt`
2. **backend_Dockerfile.txt** → `groundrag/backend/Dockerfile`
3. **backend_env_example.txt** → `groundrag/backend/.env.example`
4. **backend_main.py** → `groundrag/backend/app/main.py`
5. **backend_settings.py** → `groundrag/backend/app/config/settings.py`
6. **qdrant_client.py** → `groundrag/backend/app/db/qdrant_client.py`
7. **request_models.py** → `groundrag/backend/app/models/request_models.py`
8. **response_models.py** → `groundrag/backend/app/models/response_models.py`
9. **masa_agent.py** → `groundrag/backend/app/agents/masa_agent.py`
10. **retriever_tool.py** → `groundrag/backend/app/agents/tools/retriever_tool.py`
11. **image_tool.py** → `groundrag/backend/app/agents/tools/image_tool.py`
12. **audio_tool.py** → `groundrag/backend/app/agents/tools/audio_tool.py`
13. **video_tool.py** → `groundrag/backend/app/agents/tools/video_tool.py`
14. **upload_api.py** → `groundrag/backend/app/api/v1/upload.py`
15. **chat_api.py** → `groundrag/backend/app/api/v1/chat.py`
16. **sources_api.py** → `groundrag/backend/app/api/v1/sources.py`
17. **api_v1_init.py** → `groundrag/backend/app/api/v1/__init__.py`

Frontend Files

1. **frontend_requirements.txt** → `groundrag/frontend/requirements.txt`
2. **frontend_Dockerfile.txt** → `groundrag/frontend/Dockerfile`
3. **frontend_env_example.txt** → `groundrag/frontend/.env.example`

init.py Files

Create **empty** `__init__.py` files in these directories:

- `backend/app/__init__.py`
- `backend/app/config/__init__.py`

- backend/app/api/__init__.py
- backend/app/agents/__init__.py
- backend/app/agents/tools/__init__.py
- backend/app/services/__init__.py
- backend/app/models/__init__.py
- backend/app/db/__init__.py
- backend/app/utils/__init__.py
- frontend/components/__init__.py
- frontend/services/__init__.py
- frontend/utils/__init__.py

Quick command to create all init.py files:

```
# In groundrag/backend/
touch app/__init__.py
touch app/config/__init__.py
touch app/api/__init__.py
touch app/api/v1/__init__.py
touch app/agents/__init__.py
touch app/agents/tools/__init__.py
touch app/services/__init__.py
touch app/models/__init__.py
touch app/db/__init__.py
touch app/utils/__init__.py

# In groundrag/frontend/
touch components/__init__.py
touch services/__init__.py
touch utils/__init__.py
```

Installation Steps {#installation}

Step 3: Install Dependencies

Backend Installation

```
cd groundrag/backend

# Create virtual environment
python -m venv venv

# Activate virtual environment
# On Windows:
venv\\Scripts\\activate
# On Mac/Linux:
source venv/bin/activate
```

```
# Install dependencies
pip install -r requirements.txt
```

Frontend Installation

```
cd groundrag/frontend

# Create virtual environment
python -m venv venv

# Activate virtual environment
# On Windows:
venv\\Scripts\\activate
# On Mac/Linux:
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt
```

Configuration {#configuration}

Step 4: Configure Environment Variables

Backend Configuration

1. Copy `.env.example` to `.env`:

```
cd groundrag/backend
cp .env.example .env
```

2. Edit `.env` file and add your Google API key:

```
# Open .env in your text editor
# Add your actual API key:
GOOGLE_API_KEY=your_actual_google_api_key_here

# Other settings can remain as default
```

Important: Get your Google API key from:

- https://makersuite.google.com/app/apikey
- OR https://console.cloud.google.com/

Frontend Configuration

1. Copy `.env.example` to `.env`:

```
cd groundrag/frontend
cp .env.example .env
```

2. The default settings should work:

```
BACKEND_API_URL=http://localhost:8000
```

Running the Application {#running}

Option 1: Run with Docker (RECOMMENDED)

This is the easiest way to run the entire stack.

Step 5A: Start All Services with Docker

```
# From groundrag/ root directory
docker-compose up --build
```

This will start:

- ✅ Qdrant (port 6333)
- ✅ Backend API (port 8000)
- ✅ Frontend UI (port 8501)

Wait for all services to start (about 1-2 minutes)

Access the application:

- **Frontend UI:** http://localhost:8501
- **Backend API Docs:** http://localhost:8000/docs
- **Qdrant Dashboard:** http://localhost:6333/dashboard

To stop all services:

```
docker-compose down
```

Option 2: Run Locally (Development Mode)

For active development, run services separately.

Step 5B: Start Services Manually

Terminal 1 - Start Qdrant:

```
docker run -p 6333:6333 -p 6334:6334 qdrant/qdrant:latest
```

Terminal 2 - Start Backend:

```
cd groundrag/backend
source venv/bin/activate  # or venv\\Scripts\\activate on Windows
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

Terminal 3 - Start Frontend:

```
cd groundrag/frontend
source venv/bin/activate  # or venv\\Scripts\\activate on Windows
streamlit run app.py --server.port 8501
```

Access URLs:

- **Frontend:** http://localhost:8501
- **Backend API:** http://localhost:8000/docs
- **Qdrant:** http://localhost:6333/dashboard

Testing the System {#testing}

Step 6: Verify Installation

1. Check Backend Health

Open browser: http://localhost:8000/health

Expected response:

```
{
  "status": "healthy",
  "version": "1.0.0",
  "qdrant_connected": true
}
```

2. Check API Documentation

Open browser: http://localhost:8000/docs

You should see interactive Swagger documentation.

3. Test Frontend

Open browser: http://localhost:8501

You should see the GroundRAG interface with:

- Left panel: Upload section
- Right panel: Chat interface

4. Upload Test Document

1. Create a simple test PDF or use any PDF file
2. Click "Upload Document" in the left panel
3. Select your PDF file
4. Click "Process Document"
5. Wait for success message: "✅ Uploaded: filename.pdf"

5. Test Chat Query

1. Select your uploaded document (checkbox)
2. Type a question in the chat input
3. Press Enter
4. You should see:
    - Your question
    - AI response with citations

Troubleshooting {#troubleshooting}

Common Issues and Solutions

Issue 1: "GOOGLE_API_KEY not set"

Solution:

- Check `backend/.env` file
- Ensure `GOOGLE_API_KEY=your_actual_key` (no quotes)
- Restart backend service

Issue 2: "Cannot connect to Qdrant"

Solution:

- Verify Qdrant is running: `docker ps | grep qdrant`
- Check Qdrant at: http://localhost:6333/dashboard
- If not running: `docker run -p 6333:6333 qdrant/qdrant:latest`

Issue 3: "Module not found" errors

Solution:

- Ensure virtual environment is activated
- Reinstall requirements: `pip install -r requirements.txt`
- Check Python version: `python --version` (should be 3.11+)

Issue 4: Backend won't start

Solution:

- Check for port conflicts: `lsof -i :8000` (Mac/Linux) or `netstat -ano | findstr :8000` (Windows)
- Kill conflicting process
- Check logs for errors

Issue 5: Frontend can't connect to backend

Solution:

- Verify backend is running: http://localhost:8000/health
- Check `frontend/.env`: `BACKEND_API_URL=http://localhost:8000`
- Check CORS settings in `backend/app/config/settings.py`

Issue 6: Import errors for smolagents

Solution:

```
pip install --upgrade smolagents
pip install litellm
```

Next Steps {#next-steps}

After Successful Installation

1. Test Multi-Source Features

Upload Different Source Types:

- ✅ PDF documents
- ✅ Web page URLs
- ✅ YouTube video URLs
- ✅ DOCX files
- ✅ PowerPoint files

Example URLs to test:

- Web: `https://en.wikipedia.org/wiki/Artificial_intelligence`
- Video: `https://www.youtube.com/watch?v=aircAruvnKk` (Neural Networks)

2. Test Advanced Features

Multi-Source Chat:

1. Upload multiple documents
2. Select all sources (checkboxes)
3. Ask questions that span multiple sources

Citation Testing:

1. Ask specific questions
2. Check if answers include `[Source: ..., Page: X]`
3. Verify citations are accurate

3. Customize the System

Modify Agent Behavior:

- Edit `backend/app/agents/masa_agent.py`
- Change `SYSTEM_PROMPT` to customize responses

Adjust Chunking:

- Edit `backend/app/config/settings.py`
- Modify `CHUNK_SIZE` and `CHUNK_OVERLAP`

Change UI:

- Edit `frontend/components/*.py`
- Customize Streamlit layout

4. Add Features

Planned Enhancements:

- ✅ Advanced citation display (expandable sections)
- ✅ Source filtering
- ✅ Chat history persistence
- ✅ Export conversations
- ✅ Multi-language support

Development Workflow

Making Changes

Backend Changes

1. Edit files in `backend/app/`
2. Backend auto-reloads (if using `--reload`)
3. Test at http://localhost:8000/docs

Frontend Changes

1. Edit files in `frontend/`
2. Streamlit auto-reloads
3. Refresh browser at http://localhost:8501

Database Reset

To clear all data:

```
docker-compose down -v
docker-compose up
```

Production Deployment (Future)

When ready for production:

1. **Set strong API keys**
2. **Use Qdrant Cloud** (instead of local Docker)
3. **Deploy backend** (AWS, Google Cloud, Azure)
4. **Deploy frontend** (Streamlit Cloud, Heroku)
5. **Set up CI/CD** (GitHub Actions)

6. **Add monitoring** (Prometheus, Grafana)
7. **Enable HTTPS**
8. **Add authentication**

Quick Reference Commands

Start Everything (Docker)

```
docker-compose up
```

Start Backend Only (Local)

```
cd backend
source venv/bin/activate
uvicorn app.main:app --reload
```

Start Frontend Only (Local)

```
cd frontend
source venv/bin/activate
streamlit run app.py
```

View Logs (Docker)

```
docker-compose logs -f backend
docker-compose logs -f frontend
docker-compose logs -f qdrant
```

Stop All Services

```
docker-compose down
```

Reset Everything

```
docker-compose down -v
docker-compose up --build
```

Support & Resources

Documentation

- FastAPI: https://fastapi.tiangolo.com/
- Streamlit: https://docs.streamlit.io/
- smolagents: https://huggingface.co/docs/smolagents
- Qdrant: https://qdrant.tech/documentation/
- Gemini API: https://ai.google.dev/

Getting Help

- Check logs for errors
- Review troubleshooting section
- Verify all prerequisites are met
- Check API key permissions

Installation Complete!

You now have a fully functional GroundRAG system with:

- ✅ Multi-source document ingestion
- ✅ Agentic AI with smolagents
- ✅ Vector search with Qdrant
- ✅ Multi-modal understanding (text, images, audio, video)
- ✅ Complete citation grounding
- ✅ Streamlit web interface

Next: Start uploading your documents and chatting with your sources!