PineScript Expert Setup Guide

AI Agent System - Phase 2.0 Implementation

February 12, 2026

Overview

This document chronicles the complete setup process for creating a PineScript Expert AI agent using Ollama (free local LLM) instead of OpenAI. The setup included installing dependencies, configuring the environment, troubleshooting multiple issues, and successfully launching a working Streamlit interface.

System Requirements

- macOS (Apple Silicon - M-series chip)
- Homebrew package manager
- Python 3.14
- Docker Desktop (running)
- Git (for version control)

Step 1: Install Ollama

Ollama is a free, local LLM runtime that eliminates the need for OpenAI API keys.

```
brew install ollama
```

Start Ollama as a background service:

```
brew services start ollama
```

Pull the Llama 3.2 model (~2GB download):

```
ollama pull llama3.2
```

Verify installation:

```
ollama list
```

Expected output: llama3.2:latest (2.0 GB)

Step 2: Create Project from Template

```
cd ~/agents
python3 create_project.py pinescript 54320
```

This creates: ~/agents/projects/pinescript-expert/

Step 3: Configure Environment

Navigate to project:

```
cd ~/agents/projects/pinescript-expert
```

Edit .env file:

```
nano .env
```

Add these lines:

```
PROJECT_NAME=pinescript-expert
PORT=54320
DATABASE_URL=postgresql://postgres:postgres@localhost:54320/pinescript-expert
OLLAMA_BASE_URL=http://localhost:11434/v1
```

Note: No OpenAI API key needed when using Ollama!

Step 4: Update Configuration Files

Edit configs/project.yaml:

```
nano configs/project.yaml
```

Change the agent model setting:

```
model: ollama:llama3.2:latest
```

Step 5: Set Up Virtual Environment

Python 3.14 requires virtual environments to prevent system package conflicts.

Create virtual environment:

```
python3 -m venv venv
```

Activate it (do this every time you work on the project):

```
source venv/bin/activate
```

Your prompt will show (venv) when activated.

Step 6: Install Dependencies

CRITICAL: Use pip3, not pip, on macOS:

```
pip3 install -r requirements.txt
```

Upgrade pip (optional but recommended):

```
pip install --upgrade pip
```

Step 7: Start Database

The Docker warning about 'version' is harmless and can be ignored.

```
docker-compose up -d
```

Initialize database tables:

```
python3 scripts/init_db.py
```

Expected output: 'Database initialized successfully'

Troubleshooting Issues Encountered

Issue 1: OLLAMA_BASE_URL Not Set

Error:
```
UserError: Set the `OLLAMA_BASE_URL` environment variable
```

Solution: Added to .env file:
```
OLLAMA_BASE_URL=http://localhost:11434/v1
```

Note: The /v1 endpoint is required for OpenAI-compatible API mode.

Issue 2: Model Not Found (404 Error)

Error:
```
status_code: 404, model_name: llama3.2
```

Debugging steps:
• Verified Ollama was running: lsof -i :11434
• Checked available models: ollama list
• Tested model directly: ollama run llama3.2
• Confirmed API endpoint: curl http://localhost:11434/api/tags

Root cause: Missing /v1 in OLLAMA_BASE_URL. Pydantic AI requires OpenAI-compatible endpoint.

Issue 3: AgentRunResult Attribute Error

Error:
```
'AgentRunResult' object has no attribute 'data'
```

Solution: Updated TWO files:

1. agents/base.py - Changed return statement:
```
async def run(self, query):
    result = await self.agent.run(query)
    return result.output  # Changed from result.data
```

2. controllers/controller.py - Fixed to use BaseAgent's run method properly
The controller was bypassing the BaseAgent wrapper and calling Pydantic AI directly.

Step 8: Launch Application

```
streamlit run interfaces/streamlit_app.py
```

The application should:
• Open browser automatically at http://localhost:8501
• Display 'pinescript-expert' title with robot icon
• Show chat input: 'Ask me anything...'
• Respond to queries using Ollama's Llama 3.2 model

Daily Workflow

Every time you return to work on this project:

1. Navigate to project: cd ~/agents/projects/pinescript-expert
2. Activate virtual environment: source venv/bin/activate
3. Start database (if not running): docker-compose up -d
4. Launch Streamlit: streamlit run interfaces/streamlit_app.py
5. Stop Streamlit: Press Ctrl+C in terminal
6. Deactivate venv when done: deactivate

Template Updates Made

Updated the base template for future projects:

~/agents/templates/project_template/.env.example:

```
OLLAMA_BASE_URL=http://localhost:11434/v1
```

~/agents/templates/project_template/agents/base.py:

```
model = self.config.get('model', 'ollama:llama3.2')
# Changed fallback from openai:gpt-4o-mini

return result.output  # Changed from result.data
```

Key Learnings

• Always use pip3 (not pip) on macOS for Python package installation
• Virtual environments are mandatory in Python 3.14+ to prevent system conflicts
• Ollama requires /v1 endpoint suffix for OpenAI-compatible API mode
• Use 'ollama list' to verify models before running applications
• Pydantic AI changed from result.data to result.output in recent versions
• Docker Compose 'version' field is deprecated but harmless (warning can be ignored)
• Always activate venv before running Python commands: source venv/bin/activate
• Press Ctrl+C to stop Streamlit and most terminal processes

Useful Verification Commands

Check Ollama status:
```
brew services list | grep ollama
```

List downloaded models:
```
ollama list
```

Test Ollama API:
```
curl http://localhost:11434/api/tags
```

Check port 11434:
```
lsof -i :11434
```

Test model directly:
```
ollama run llama3.2
```

Check Docker containers:
```
docker ps
```

Check Python version:
```
python3 --version
```

List pip packages in venv:
```
pip list
```

Next Steps: Phase 2.0 Milestones

Now that Phase 1.5 is complete with a working basic agent, proceed to Phase 2.0:

• M2.2: Implement PineScript-specific agent with domain expertise
• M2.3: Implement PineScript controller with orchestration logic
• M2.4: Update Streamlit interface to use domain-specific components
• M2.5: Test with PineScript queries and validate responses

This completes the Phase 1.5 → Phase 2.0 setup. The system is now ready for domain-specific implementation.