AI VOID FIELD GUIDE

Step-by-Step Tutorials

Focused, practical tutorials that teach you how to accomplish specific tasks step by step. Learn by doing with clear instructions and working code examples.

AI VOID

2026-05-04 17:41 IST

Contents

CHAPTER 01

How to Integrate VS Code with Ollama for Local AI Assistance: Step-by-Step Guide

Introduction

This tutorial will guide you through setting up a powerful, private, and cost-free AI coding assistant directly within your Visual Studio Code environment. By integrating Ollama with the Continue VS Code extension, you'll be able to run large language models (LLMs) locally on your machine. This setup allows for code generation, completion, debugging assistance, and refactoring without relying on external APIs, ensuring complete privacy for your code and eliminating API costs.

You'll learn how to install Ollama, download a suitable coding-focused LLM, install the Continue extension in VS Code, and configure it to use your local Ollama instance. Finally, we'll demonstrate practical examples of using the AI assistant for common development tasks.

What you'll accomplish: * Install and run Ollama on your local machine. * Download a code-specific large language model (LLM). * Install and configure the Continue extension in Visual Studio Code. * Use your local AI assistant for code generation, explanation, and refactoring.

Why it's useful:

- **Privacy:** Your code never leaves your machine.
- **Cost-Free:** No API usage fees.
- **Speed:** Local models often provide faster responses.
- **No Limits:** No token limits or usage restrictions.
- **Offline Capability:** Work with AI assistance even without an internet connection (after initial setup).

Time estimate: Approximately 30-60 minutes, depending on your internet speed and system performance for model downloads.

Prerequisites

Before you begin, ensure you have the following:

- ✅ **Operating System:** Windows 10/11, macOS, or Linux.
- ✅ **Hardware:** A machine with at least 8GB of RAM (16GB+ recommended for larger models) and a modern CPU. A dedicated GPU (NVIDIA or AMD) with good VRAM will significantly speed up inference, but it's not strictly required for smaller models.
- ✅ **Visual Studio Code:** Version 1.80 or newer. Download from code.visualstudio.com.
- ✅ **Internet Connection:** Required for initial Ollama and model downloads, and VS Code extension installation.
- ✅ **Basic Command Line Interface (CLI) Knowledge:** Familiarity with running commands in your terminal.

Step-by-Step Instructions

Step 1: Install Ollama

Ollama is an open-source tool that allows you to run large language models locally. This step involves downloading and installing the Ollama server on your machine.

Explanation: Ollama handles the heavy lifting of managing and running LLMs. It provides an API that the VS Code Continue extension will use to communicate with the models.

1. **Download Ollama:** Open your web browser and navigate to the official Ollama website: https://ollama.com/download.
2. **Select your Operating System:** Download the appropriate installer for Windows, macOS, or Linux.
3. **Run the Installer:**

- **Windows:** Run the `.exe` installer and follow the on-screen prompts.
- **macOS:** Open the `.dmg` file and drag the Ollama application to your Applications folder. Then, open Ollama from your Applications folder. It will start running in the background.

- **Linux:** Open your terminal and run the following command: `bash curl -fsSL https://ollama.com/install.sh | sh` This script will install Ollama and set it up as a system service.

Verify it worked: After installation, open your terminal or command prompt and run:

```
ollama --version
```

You should see output similar to this, indicating Ollama is installed and running:

```
ollama version is 0.1.32
```

Then, try listing available models (initially, this list will be empty):

```
ollama list
```

```
NAME    ID  SIZE    MODIFIED
```

Troubleshooting:

- **`ollama: command not found` (Linux/macOS):** Ensure Ollama is added to your system's PATH. On Linux, the install script usually handles this. On macOS, ensure you've opened the Ollama app at least once. Try restarting your terminal.
- **Installation failed:** Re-download the installer and try again. Check your system's minimum requirements.

Step 2: Pull an AI Model with Ollama

Now that Ollama is installed, you need to download an actual AI model to use. For coding assistance, `codellama` is an excellent choice.

Explanation: Ollama itself is just the runner; you need a model (the "brain") to perform AI tasks. `codellama` is specifically trained for programming-related tasks.

1. **Open your terminal** or command prompt.
2. **Pull the `codellama` model:** Execute the following command. This will download the model, which can take several minutes depending on your internet speed and the model size. `bash ollama run codellama` If you want a specific tag (e.g., a smaller 7B parameter model), you can specify it:

`bash ollama run codellama:7b` The command will start downloading the model. You'll see a progress indicator. Once downloaded, it will immediately enter an interactive chat session with the model. You can type `bye` or press `Ctrl+D` to exit this session.

Verify it worked: After the download completes and you exit the interactive session, list the installed models again:

```
ollama list
```

You should now see `codellama` (or `codellama:7b`) in the list:

```
NAME            ID              SIZE    MODIFIED
codellama:latest    ab123c4d5e6f    3.8 GB  2 minutes ago
```

Troubleshooting:

- **`Error: pull model failed`:** Check your internet connection. Ensure there's enough disk space on your machine. Sometimes, a temporary network issue can cause this; try the command again.
- **Stuck on "downloading...":** This usually means a slow internet connection. Be patient. If it completely freezes, cancel with `Ctrl+C` and try again.

Step 3: Install Visual Studio Code

If you don't already have Visual Studio Code installed, download and install it now.

Explanation: VS Code will be our integrated development environment (IDE) where we'll write code and interact with the AI assistant.

1. **Download VS Code:** Go to https://code.visualstudio.com/ and download the installer for your operating system.
2. **Run the Installer:** Follow the instructions to install VS Code. For most users, the default options are sufficient.

Verify it worked: Launch Visual Studio Code. You should see the welcome screen or your last opened workspace.

Troubleshooting:

- **VS Code not opening:** Try restarting your computer. If issues persist, refer to the official VS Code troubleshooting guide for your operating system.

Step 4: Install the VS Code Continue Extension

The Continue extension acts as the bridge between VS Code and your local Ollama instance.

Explanation: Continue provides the user interface and logic within VS Code to send your code and prompts to the Ollama server and display the AI's responses.

1. **Open VS Code.**
2. **Go to the Extensions view:** Click on the Extensions icon in the Activity Bar on the side of the window (it looks like four squares, one of which is separated) or press `Ctrl+Shift+X` (Windows/Linux) / `Cmd+Shift+X` (macOS).
3. **Search for "Continue":** In the search bar at the top of the Extensions view, type `Continue`.
4. **Install the extension:** Locate the "Continue" extension by **Continue Dev** and click the **Install** button.

Verify it worked: After installation, a new Continue icon (a circle with a lightning bolt) should appear in the Activity Bar on the left side of your VS Code window. Click this icon to open the Continue sidebar.

Troubleshooting:

- **Extension not found:** Double-check your spelling. Ensure you have an active internet connection.
- **Installation errors:** Restart VS Code and try installing again. Check the VS Code Output panel (`View > Output`, then select "Log (Extension Host)" from the dropdown) for more detailed error messages.

Step 5: Configure Continue to Use Ollama

Now, you need to tell the Continue extension to use your locally running Ollama server and the `codellama` model you pulled.

Explanation: By default, Continue might try to use cloud-based AI services. We need to explicitly configure it to connect to your local Ollama instance.

1. **Open the Continue sidebar:** Click the Continue icon in the Activity Bar.
2. **Open Continue settings:** In the Continue sidebar, click the gear icon (⚙️) at the top, or click the "Configure Continue" button if it's visible. This will open a `config.json` file. If prompted to create a new config, proceed.

3. **Edit `config.json`:** Replace the existing content of `config.json` with the following configuration. This tells Continue to use Ollama and specifically the `codellama` model.

   `json { "models": [ { "name": "codellama", "provider": "ollama", "model": "codellama", "temperature": 0.5, "topP": 0.9, "maxTokens": 1024 } ], "defaultModel": "codellama" }` **Note:** If you pulled `codellama:7b`, change `"model": "codellama"` to `"model": "codellama:7b"`.

4. **Save the `config.json` file:** Press `Ctrl+S` (Windows/Linux) or `Cmd+S` (macOS).

Verify it worked: In the Continue sidebar, you should see "codellama" selected as the active model. The status at the bottom of the sidebar should indicate "Ready" or a similar positive status. If there are errors, they will usually be displayed here.

Troubleshooting:

- **"Model not found" or "Connection refused" in Continue sidebar:**
  - Ensure Ollama is running in the background (check your system's process monitor, or try `ollama list` in the terminal).
  - Verify the `model` name in `config.json` exactly matches the name you see when you run `ollama list`.
  - Restart VS Code.
  - Check if any firewall is blocking communication to `localhost:11434` (Ollama's default port).
- **`config.json` syntax errors:** Ensure your JSON is valid. Use an online JSON validator if unsure. A missing comma or bracket can cause issues.

Testing the Complete Setup

Now that everything is configured, let's test your local AI coding assistant!

1. **Open a new or existing code file** in VS Code (e.g., a `.py`, `.js`, or `.ts` file).
2. **Open the Continue sidebar.**
3. **Provide a prompt:** In the input box at the bottom of the Continue sidebar, type a request.

Example 1: Generate a Python function Type: `Write a Python function to calculate the factorial of a number recursively.` Press Enter.

Expected Results: The AI should generate code in the Continue chat panel, similar to this:

```python def factorial_recursive(n): """ Calculates the factorial of a number recursively. """ if n == 0: return 1 else: return n * factorial_recursive(n - 1)

# Example usage:

# result = factorial_recursive(5)

# print(result) # Output: 120

```
You can then click "Insert into new file" or copy-paste the code into your editor.

Example 2: Explain selected code 1. Select a block of code in your editor. 2. In the Continue sidebar, type: `Explain this code.` Press Enter.

Expected Results: The AI should provide an explanation of the selected code.

Example 3: Debug an error 1. Imagine you have a Python error: `TypeError: can only concatenate str to str` 2. Copy the error message and the relevant code snippet. 3. In the Continue sidebar, type: ```
I'm getting this error: TypeError: can only concatenate str to str. Here's my code: def greet(name): return "Hello, " + name + 5 # Intentional error

```
greet("Alice")

What's wrong and how do I fix it?
```
Press Enter.
```

**Expected Results:** The AI should identify the type mismatch and suggest converting `5` to a string or removing it if it's not intended to be part of the greeting.
```

Troubleshooting Guide

Here's a consolidated list of common issues and their solutions:

- **Ollama server not running:**
- **Symptom:** Continue sidebar shows "Connection refused" or "Ollama not running."
- **Solution:** Ensure Ollama is running. On macOS, open the Ollama application. On Windows, check your system tray for the Ollama icon. On Linux, verify the `ollama` service is active (`systemctl status ollama`). If not, start it (`systemctl start ollama`).
- **Model not found in Ollama:**
- **Symptom:** Continue reports "Model 'codellama' not found."
- **Solution:** Verify the model is downloaded by running `ollama list` in your terminal. If it's not there, pull it using `ollama run codellama` (or the specific tag you prefer). Ensure the `model` name in your `config.json` exactly matches the name from `ollama list`.
- **Continue sidebar is empty or unresponsive:**
- **Symptom:** The Continue chat panel doesn't appear, or typing a prompt yields no response.
- **Solution:** 1. Restart VS Code. 2. Check the VS Code Output panel (`View > Output`) for "Continue" or "Log (Extension Host)" for error messages. 3. Re-check your `config.json` for syntax errors. 4. Ensure Ollama is running and accessible.
- **Slow AI responses:**
- **Symptom:** It takes a very long time for the AI to generate a response.
- **Solution:** 1. **Hardware:** LLMs are resource-intensive. Ensure you have sufficient RAM. A dedicated GPU with VRAM will drastically improve performance. 2. **Model Size:** You might be using a larger model (e.g., `codellama:34b`). Consider using a smaller, faster model like `codellama:7b` for general use. You can pull smaller models and update your `config.json` accordingly. 3. **Other processes:** Close other resource-intensive applications running on your machine.
- **Firewall blocking Ollama:**
- **Symptom:** "Connection refused" errors even when Ollama is running.

- **Solution:** Your system's firewall might be blocking the connection to `localhost:11434`. Temporarily disable your firewall or create an exception for Ollama.
- **Inaccurate or irrelevant AI responses:**
- **Symptom:** The AI's suggestions are not helpful or are completely wrong.
- **Solution:** 1. **Refine your prompt:** Be more specific and provide more context in your requests. 2. **Model Choice:** While `codellama` is good, different models excel at different tasks. Explore other models on Ollama's library (e.g., `deepseek-coder`, `mistral`) and test them.

Next Steps

Congratulations! You now have a fully functional local AI coding assistant in VS Code. Here are some ideas for what to explore next:

- **Experiment with other Ollama models:** Check out the Ollama library for other models like `llama2`, `mistral`, `deepseek-coder`, or `phi3`. You can pull them using `ollama run <model_name>` and then update your `config.json` to switch between them.
- **Explore advanced Continue features:** Continue offers features like multi-line edits, custom commands, and context awareness (feeding specific files or documentation to the LLM). Read the Continue documentation for more.
- **Fine-tuning models:** For advanced users, you can explore fine-tuning smaller models with your own codebase to make them even more relevant to your specific projects.
- **Integrate with other tools:** Ollama also provides an API that can be integrated into other applications or scripts.

References

- **Ollama Official Website:** https://ollama.com/
- **Continue VS Code Extension:** https://continue.dev/
- **Visual Studio Code:** https://code.visualstudio.com/
- **Ollama Model Library:** https://ollama.com/library

Transparency Note

This tutorial was created by an AI expert based on current best practices and information available as of 2026-04-09. While every effort has been made to ensure accuracy and functionality, software environments and tools evolve rapidly. If you encounter discrepancies or issues, please refer to the official documentation of Ollama, Visual Studio Code, and the Continue extension for the most up-to-date information.