PHASE 0

Environment Setup Guide

Complete Beginner's Guide to Docker, PostgreSQL, and Python Development

Last Updated: February 2026

TABLE OF CONTENTS

SECTION 1: WHAT YOU'RE BUILDING

Before installing anything, let's understand what each tool does and why you need it.

Tool	What It Does	Why You Need It
Docker	Runs applications in isolated containers	Runs PostgreSQL database without installing it globally
PostgreSQL	Relational database with vector support	Stores documentation chunks and embeddings for RAG
Python 3	Programming language	Runs your agent code and AI models
Git	Version control system	Tracks changes to your code
Homebrew	Package manager for Mac	Easiest way to install Docker, Python, Git

The Big Picture:

Your AI agent will run in Python. It needs a database (PostgreSQL) to store documentation. Docker makes running PostgreSQL easy - you just start/stop a container instead of installing a full database server on your Mac.

SECTION 2: INSTALL HOMEBREW

Homebrew is the easiest way to install developer tools on Mac. Think of it like an App Store for command-line tools.

Step 1: Open Terminal

- Press **Cmd + Space** to open Spotlight
- Type 'Terminal' and press Enter
- You'll see a window with a command prompt (something like `username@macbook ~ %`)

Step 2: Check if Homebrew is Already Installed

Type this command and press Enter:

```
brew --version
```

✓ Homebrew is already installed! Skip to Section 3.

If you see 'command not found':
Continue to Step 3 to install Homebrew.

Step 3: Install Homebrew

Copy this entire command, paste it into Terminal, and press Enter:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

What will happen:

- The installer will ask for your Mac password (the one you use to log in)
- It will download and install Homebrew (takes 2-5 minutes)
- You'll see a lot of text scrolling - this is normal
- When done, you'll see 'Installation successful!'

■■ IMPORTANT: After installation, Homebrew may show instructions to add it to your PATH. Copy and run those commands!

Step 4: Verify Installation

Close Terminal and open a new Terminal window, then run:

```
brew --version
```

You should see: `Homebrew 4.x.x`

SECTION 3: INSTALL DOCKER DESKTOP

Docker Desktop is a graphical application that runs Docker containers. It's the easiest way to use Docker on Mac.

Step 1: Install Docker Desktop via Homebrew

In Terminal, run this command:

```
brew install --cask docker
```

What will happen:

- Downloads Docker Desktop (about 500MB - takes 5-10 minutes on average internet)
- Installs it in your Applications folder
- You'll see progress messages

Step 2: Launch Docker Desktop

- Open your Applications folder (Cmd + Space, type 'Applications')
- Find and double-click 'Docker'
- Docker Desktop will ask for permission - click 'OK'
- You may need to enter your Mac password
- Docker Desktop will start (look for a whale icon in your menu bar)

Step 3: Complete Docker Setup

- Docker Desktop will show a welcome screen
- You can skip the tutorial (click 'Skip tutorial')
- Accept the service agreement if prompted
- Docker Desktop will start the Docker engine (takes 30-60 seconds)
- When ready, you'll see 'Docker Desktop is running' with a green indicator

■■ Docker Desktop must be running for your database to work. The whale icon in your menu bar should NOT have a red dot.

SECTION 4: VERIFY DOCKER INSTALLATION

Let's make sure Docker is working correctly.

Step 1: Check Docker Version

In Terminal, run:

```
docker --version
```

Expected output: `Docker version 25.x.x` (or higher)

Step 2: Check Docker Compose

Run this command:

```
docker compose version
```

Expected output: `Docker Compose version v2.x.x`

Step 3: Test Docker with a Simple Container

Run this test command:

```
docker run hello-world
```

What should happen:

- Docker downloads a tiny test image
- Runs a container that prints 'Hello from Docker!'
- Shows a success message explaining what happened
- Container stops automatically

✓ If you see the success message, Docker is working perfectly!

SECTION 5: INSTALL PYTHON 3

Your Mac comes with Python 2.7 (outdated). You need Python 3.11 or higher for your AI agent.

Step 1: Check Current Python Version

Run this command:

```
python3 --version
```

✓ You're good! Skip to Section 6.

If you see Python 3.9 or lower (or 'command not found'):
Continue to Step 2.

Step 2: Install Python 3.12 via Homebrew

Run this command:

```
brew install python@3.12
```

What will happen:

- Homebrew downloads Python 3.12 (takes 3-5 minutes)
- Installs Python and pip (Python's package installer)
- You'll see lots of installation messages

Step 3: Verify Python Installation

Close Terminal, open a new Terminal window, and run:

```
python3 --version
```

Expected: `Python 3.12.x`

Step 4: Verify pip (Python Package Installer)

Run:

```
pip3 --version
```

Expected: `pip 24.x (python 3.12)`

SECTION 6: INSTALL GIT

Git tracks changes to your code so you can save versions and recover from mistakes.

Step 1: Check if Git is Installed

Run:

```
git --version
```

✓ Git is already installed! Skip to Section 7.

If you see 'command not found':
Continue to Step 2.

Step 2: Install Git

Run:

```
brew install git
```

Step 3: Verify Installation

Close and reopen Terminal, then run:

```
git --version
```

Expected: `git version 2.x.x`

Step 4: Configure Git (First Time Only)

Tell Git who you are (replace with your actual name and email):

```
git config --global user.name "Your Name"
```

```
git config --global user.email "your.email@example.com"
```

SECTION 7: CREATE YOUR AGENT SYSTEM DIRECTORY

Set up the folder structure for your AI agent system.

Step 1: Navigate to Home Directory

In Terminal, run:

```
cd ~
```

This takes you to your home folder (usually /Users/yourname)

Step 2: Create Agent System Folder

Run:

```
mkdir -p agents/templates
```

This creates: ~/agents/ and ~/agents/templates/

Step 3: Navigate to Agent Directory

Run:

```
cd ~/agents
```

Step 4: Verify Directory Creation

Run:

```
pwd
```

Expected output: `/Users/yourname/agents`

SECTION 8: TEST YOUR COMPLETE SETUP

Let's verify everything is working together.

Checklist - Run Each Command:

Tool	Command	Expected Output
Homebrew	`brew --version`	`Homebrew 4.x.x`
Docker	`docker --version`	`Docker version 25.x.x`
Docker Compose	`docker compose version`	`Docker Compose version v2.x.x`
Python	`python3 --version`	`Python 3.12.x`
Pip	`pip3 --version`	`pip 24.x (python 3.12)`
Git	`git --version`	`git version 2.x.x`

Final Test: Run a PostgreSQL Container

This tests that Docker can run your database. Run this command:

```
docker run -d -p 5432:5432 -e POSTGRES_PASSWORD=test --name test-postgres
pgvector/pgvector:pg16
```

What happens:

- Docker downloads the PostgreSQL image with vector extension (~200MB)
- Starts a database container named 'test-postgres'
- Database runs on port 5432
- Returns a long container ID (this is normal)

Verify it's running:

```
docker ps
```

You should see 'test-postgres' in the list of running containers.

Stop and remove the test container:

```
docker stop test-postgres
```

```
docker rm test-postgres
```

✓ If all these tests passed, your environment is ready for Phase 2!

SECTION 9: TROUBLESHOOTING

Problem: 'brew command not found'

- Close Terminal completely (Cmd + Q)
- Open a new Terminal window
- If still not found, Homebrew wasn't added to PATH. Run: `echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> ~/.zprofile`
- Then run: `source ~/.zprofile`

Problem: Docker Desktop won't start

- Quit Docker Desktop completely
- Open Activity Monitor (search in Spotlight)
- Search for 'Docker' and force quit any Docker processes
- Restart your Mac
- Launch Docker Desktop again

Problem: 'docker: command not found' but Docker Desktop is running

- Open Docker Desktop settings (gear icon)
- Go to 'Advanced' or 'Resources'
- Enable 'Use Docker CLI'
- Restart Terminal

Problem: 'Permission denied' errors

- For Docker: Make sure Docker Desktop is running (whale icon in menu bar)
- For file operations: Use `sudo` before the command (will ask for password)
- For git: Run `git config --global --list` to verify your settings

Problem: Port 5432 already in use

- Something else is using the PostgreSQL port
- Run: `lsof -i :5432` to see what's using it
- Either stop that service or use a different port in your project (like 54320)

APPENDIX A: UNDERSTANDING KEY CONCEPTS

What is a Container?

Think of a container like a lightweight virtual machine. It packages an application (like PostgreSQL) with everything it needs to run, isolated from the rest of your system. You can start, stop, or delete containers without affecting your Mac.

What is Docker Compose?

Docker Compose is a tool for defining multi-container applications using a YAML file. Instead of typing long docker commands, you write a docker-compose.yml file once, then just run 'docker compose up' to start everything.

What is pgvector?

pgvector is an extension for PostgreSQL that adds support for vector embeddings. This lets you store AI-generated embeddings (numerical representations of text) and search for similar content using cosine similarity. Essential for RAG (Retrieval-Augmented Generation) systems.

What is a Virtual Environment?

A Python virtual environment is an isolated Python installation. Each project gets its own environment with its own packages, preventing conflicts. You'll create one for each AI agent project.

Port Numbers Explained

Port numbers are like apartment numbers for network services. Port 5432 is PostgreSQL's standard port. When you run multiple databases, each needs its own port (5432, 54320, 54321, etc.). The port is how your Python code finds the database.

Common Terminal Commands

Command	What It Does
`pwd`	Print Working Directory - shows where you are
`ls`	List files in current directory
`cd <path>`	Change Directory - move to a different folder
`cd ~`	Go to your home directory
`mkdir <name>`	Make Directory - create a new folder

`rm <file>`	Remove file (CAREFUL - no trash bin!)
`docker ps`	List running Docker containers
`docker logs <name>`	View logs from a container

YOU'RE READY!

✓ Homebrew installed and working
✓ Docker Desktop running containers
✓ Python 3.12+ ready for AI development
✓ Git tracking your code changes
✓ Directory structure created

Next Steps:

1. Copy your template files to ~/agents/templates/project_template/
2. Copy create_project.py to ~/agents/
3. Proceed to Phase 2 of your roadmap (Basic Agent)
4. Build your first PineScript expert!

Remember: Docker Desktop must be running whenever you work on your agents. Look for the whale icon in your menu bar!

Good luck building your AI agent system! ■