Sim.AI Local Installation & Implementation Guide

AIGF Cohort 5 Fellowship

Author: Terris Foley
Date: October 2, 2025
System: Mac Studio (32GB RAM, 12 CPU cores)
Purpose: Complete guide for installing Sim.AI with Ollama for local AI agent workflows

---

Table of Contents

SIM.AI LOCAL INSTALLATION & IMPLEMENTATION GUIDE 1

AIGF COHORT 5 FELLOWSHIP 1
TABLE OF CONTENTS 1
INTRODUCTION 2
WHAT IS SIM.AI? 2
WHY USE SIM.AI? 2
PREREQUISITES 3
SYSTEM REQUIREMENTS 3
DOCKER CONFIGURATION 3
INSTALLATION PROCESS 3
STEP 1: CLONE THE REPOSITORY 3
STEP 2: CHOOSE YOUR INSTALLATION TYPE 4
STEP 3: WAIT FOR INITIALIZATION 4
TROUBLESHOOTING GUIDE 4
ISSUE 1: MEMORY ALLOCATION ERROR 4
ISSUE 2: GPU BUILD ON NON-NVIDIA SYSTEM 5
ISSUE 3: OLLAMA SERVICE NOT RUNNING 5
ISSUE 4: MODELS NOT APPEARING IN UI 6
ISSUE 5: PORT ALREADY IN USE 6
ISSUE 6: CANNOT ACCESS LOCALHOST:3000 7
POST-INSTALLATION SETUP 7
DOWNLOAD AI MODELS 7
VERIFY MODEL INSTALLATION 8
CREATE YOUR FIRST WORKFLOW 8
VERIFICATION & TESTING 8
SYSTEM HEALTH CHECK 8
UI VERIFICATION 8
TEST WORKFLOW 9
LEARNING RESOURCES 9

OFFICIAL DOCUMENTATION 9
VIDEO RESOURCES 9
COMMUNITY RESOURCES 9
KEY CONCEPTS TO MASTER 9
BEST PRACTICES 10
DEVELOPMENT WORKFLOW 10
MODEL SELECTION 10
RESOURCE MANAGEMENT 10
PRODUCTION DEPLOYMENT 10
USEFUL COMMANDS REFERENCE 10
CONTAINER MANAGEMENT 10
MODEL MANAGEMENT 11
MAINTENANCE 11
CONCLUSION 11
SUPPORT & TROUBLESHOOTING 12
APPENDIX: COMPLETE INSTALLATION SCRIPT 12

Introduction

What is Sim.AI?

Sim.AI (formerly Sim Studio) is an open-source visual workflow builder for creating AI agent workflows. It provides a Figma-like drag-and-drop canvas for building production-ready AI applications without extensive coding.

Key Features:

- Visual workflow builder with drag-and-drop interface
- Support for 100+ AI models and integrations
- Local AI model support via Ollama (no API keys required)
- Real-time team collaboration
- Multiple deployment options (API, webhooks, scheduled jobs)
- Open-source and self-hosted

Why Use Sim.AI?

- **No coding required** for basic workflows
- **Cost-effective** with local model support
- **Privacy-focused** - run entirely on your own infrastructure
- **Flexible** - works with OpenAI, Anthropic, Google, and local models
- **Production-ready** - deploy as APIs or standalone applications

---

Prerequisites

System Requirements

Minimum Requirements:

- 8GB RAM (16GB+ recommended)
- 4 CPU cores (8+ recommended)
- 20GB free disk space
- macOS, Linux, or Windows with WSL2

Software Requirements:

- Docker Desktop installed and running
- Git (for cloning repository)
- Terminal/command line access
- Web browser (Chrome, Firefox, Safari, or Edge)

Docker Configuration

CRITICAL: Before starting, configure Docker Desktop with adequate resources:

1. Open Docker Desktop
2. Navigate to Settings → Resources
3. Set the following:
    - **Memory**: 16GB (minimum 8GB)
    - **CPUs**: 8-12 (use what's available)
    - **Swap**: 2GB
    - **Disk**: 60GB+
4. Click Apply & Restart
5. Wait for Docker to fully restart

Why this matters: Insufficient memory allocation will cause build failures with "cannot allocate memory" errors.

Installation Process

Step 1: Clone the Repository

```
# Clone the Sim repository
git clone https://github.com/simstudioai/sim.git
```

```
# Navigate to project directory
cd sim
```

Step 2: Choose Your Installation Type

You have three options:

Option A: Cloud-based (with API keys)

```
docker compose -f docker-compose.prod.yml up -d
```

Access at http://localhost:3000

Option B: Local with Ollama - GPU (NVIDIA only)

```
docker compose -f docker-compose.ollama.yml --profile setup up -d
```

Option C: Local with Ollama - CPU (Recommended for Mac)

```
docker compose -f docker-compose.ollama.yml --profile cpu --profile setup up
-d
```

Step 3: Wait for Initialization

The first run will:

- Download Docker images (~5-10 minutes)
- Build application containers
- Initialize the database
- Download AI models (if using Ollama setup profile)

Monitor progress with:

```
docker compose -f docker-compose.ollama.yml logs -f
```

Press `Ctrl+C` to stop viewing logs (containers continue running).

Troubleshooting Guide

This section documents actual issues encountered and their solutions.

Issue 1: Memory Allocation Error

Symptom:

```
Error: cannot allocate memory
target simstudio: failed to solve: ResourceExhausted
```

Cause: Docker doesn't have enough RAM allocated.

Solution:

1. Stop all containers: `docker compose -f docker-compose.ollama.yml down`
2. Open Docker Desktop → Settings → Resources
3. Increase Memory to 16GB
4. Click Apply & Restart
5. Restart installation

Issue 2: GPU Build on Non-NVIDIA System

Symptom:

```
Error: could not select device driver "nvidia" with capabilities: [[gpu]]
```

Cause: Using GPU profile on Mac or non-NVIDIA systems.

Solution: Use CPU profile instead:

```
# Stop everything
docker compose -f docker-compose.ollama.yml down

# Start with CPU profile only
docker compose -f docker-compose.ollama.yml --profile cpu up -d
```

Issue 3: Ollama Service Not Running

Symptom:

```
service "ollama" is not running
```

Cause: Service name mismatch or profile not activated.

Solution:

```
# Check running services
docker compose -f docker-compose.ollama.yml ps

# Ensure CPU profile is active
docker compose -f docker-compose.ollama.yml --profile cpu up -d
```

Note: The service may be named `ollama-cpu` instead of `ollama`. Use:

```
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama list
```

Issue 4: Models Not Appearing in UI

Symptom: Ollama models don't show in the model dropdown.

Root Cause: Network alias misconfiguration preventing Sim from connecting to Ollama.

Solution: Add network alias to docker-compose.ollama.yml

1. Open the file:

```
nano docker-compose.ollama.yml
```

2. Find the `ollama-cpu` service section (around line 80-120)
3. After the `restart: unless-stopped` line, add:

```
networks:
  default:
    aliases:
      - ollama
```

Important: Maintain exact indentation (4 spaces before `networks:`).

4. Save file (Ctrl+X, Y, Enter)
5. Restart services:

```
docker compose -f docker-compose.ollama.yml down
docker compose -f docker-compose.ollama.yml --profile cpu up -d
```

6. Verify connection:

```
docker compose -f docker-compose.ollama.yml exec simstudio wget -O-
http://ollama:11434/api/tags
```

You should see JSON response with available models.

Issue 5: Port Already in Use

Symptom:

```
Error: port 3000 already in use
```

Solution:

```
# Find process using port 3000
```

```
lsof -i :3000

# Kill the process (replace PID with actual process ID)
kill -9 PID

# Or change Sim's port in docker-compose file
```

Issue 6: Cannot Access localhost:3000

Symptom: Browser shows "This site can't be reached" or "Connection refused"

Diagnosis Steps:

```
# 1. Check if containers are running
docker compose -f docker-compose.ollama.yml ps

# 2. Check container logs
docker compose -f docker-compose.ollama.yml logs simstudio

# 3. Verify port mapping
docker ps | grep 3000
```

Common Solutions:

- Wait 2-3 minutes for containers to fully start
- Check all containers show "healthy" status
- Clear browser cache and try again
- Try http://127.0.0.1:3000 instead

Post-Installation Setup

Download AI Models

After successful installation, download models for local use:

```
# Recommended starter model (fast, ~1.6GB)
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama pull
gemma2:2b

# More powerful model (~2GB)
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama pull
llama3.2:3b

# Premium models (larger downloads)
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama pull
llama3.1:8b
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama pull
mistral:7b
```

Verify Model Installation

```
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama list
```

Expected output:

```
NAME              ID              SIZE      MODIFIED
gemma2:2b         8ccf136fdd52    1.6 GB    X minutes ago
llama3.2:3b       abc123def456    2.0 GB    X minutes ago
```

Create Your First Workflow

1. Navigate to http://localhost:3000
2. Sign up for a new account (stored locally)
3. Click "New Workflow"
4. Drag an "Agent" block onto the canvas
5. Click the Agent block to configure
6. Select your model from the dropdown (e.g., gemma2:2b)
7. Add a system prompt
8. Test in the Chat panel on the right

Verification & Testing

System Health Check

Run these commands to verify everything is working:

```
# Check all services are healthy
docker compose -f docker-compose.ollama.yml ps

# Expected output: All services show "healthy" status
# - simstudio (port 3000)
# - db (port 5432)
# - ollama-cpu (port 11434)
# - realtime (port 3002)

# Test Ollama API
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama --version

# Test model availability
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama list
```

UI Verification

1. Open http://localhost:3000
2. Sign in to your account
3. Create a new workflow

4. Add an Agent block
5. Click the Agent block
6. Open the Model dropdown
7. Verify your Ollama models appear (gemma2:2b, llama3.2:3b, etc.)

Test Workflow

Create a simple test:

1. Add Agent block with gemma2:2b model
2. System prompt: "You are a helpful assistant"
3. Go to Chat panel
4. Send: "Hello, can you help me?"
5. Verify you receive a response

Learning Resources

Official Documentation

- **Main Site**: https://www.sim.ai
- **Documentation**: https://docs.sim.ai/introduction
- **Getting Started Tutorial**: https://docs.sim.ai/getting-started
- **GitHub**: https://github.com/simstudioai/sim

Video Resources

- **Official Demo**: https://youtu.be/JlCktXTY8sE
- **Additional Tutorials**: Search YouTube for "Sim Studio AI agent tutorial"

Community Resources

- **Y Combinator Page**: https://www.ycombinator.com/companies/sim
- **Hacker News Discussion**: https://news.ycombinator.com/item?id=43823096
- **Twitter/X**: @simstudioai

Key Concepts to Master

1. **Visual Workflow Builder**: Drag-and-drop interface
2. **Block Types**: Agent, API, Function, Condition, Loop, Router, Response
3. **Triggers**: Chat, API, Webhook, Scheduled
4. **Tool Integration**: 100+ pre-built integrations
5. **Deployment**: API endpoints, standalone apps
6. **Structured Output**: JSON schemas for predictable responses

---

Best Practices

Development Workflow

1. **Start Simple**: Begin with single-agent workflows
2. **Test Frequently**: Use the chat panel for rapid iteration
3. **Use Structured Output**: Define schemas for reliable data
4. **Version Control**: Save workflow versions before major changes
5. **Monitor Performance**: Check execution logs for optimization

Model Selection

- **gemma2:2b**: Fast responses, good for simple tasks, low resource usage
- **llama3.2:3b**: Balanced performance, general purpose
- **llama3.1:8b**: Higher quality, more complex reasoning, slower
- **mistral:7b**: Good for technical/coding tasks

Resource Management

- **Monitor Docker**: Keep an eye on RAM usage
- **Clean Up Regularly**:
- `docker system prune -fdocker volume prune -f`
- **Stop When Not Using**:
- `docker compose -f docker-compose.ollama.yml --profile cpu down`
- **Start Again**:
- `docker compose -f docker-compose.ollama.yml --profile cpu up -d`

Production Deployment

For production use:

- Use environment variables for sensitive data
- Set up proper authentication
- Configure SSL/TLS for HTTPS
- Implement rate limiting
- Set up monitoring and logging
- Consider using Sim's cloud hosting for managed infrastructure

---

Useful Commands Reference

Container Management

```
# Start Sim
docker compose -f docker-compose.ollama.yml --profile cpu up -d

# Stop Sim
docker compose -f docker-compose.ollama.yml --profile cpu down

# Stop and remove volumes (clean slate)
docker compose -f docker-compose.ollama.yml --profile cpu down -v

# View logs
docker compose -f docker-compose.ollama.yml logs -f

# View specific service logs
docker compose -f docker-compose.ollama.yml logs simstudio
docker compose -f docker-compose.ollama.yml logs ollama-cpu

# Restart a specific service
docker compose -f docker-compose.ollama.yml restart simstudio

# Check container status
docker compose -f docker-compose.ollama.yml ps
```

Model Management

```
# List installed models
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama list

# Download new model
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama pull
<model-name>

# Remove a model
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama rm <model-
name>

# Check running models
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama ps
```

Maintenance

```
# Clean up Docker resources
docker system prune -a -f
docker volume prune -f

# Check Docker resource usage
docker stats

# View disk usage
docker system df
```

Conclusion

You now have a fully functional local AI agent workflow builder with no external API dependencies. Sim.AI provides a powerful platform for:

- Building AI assistants and chatbots
- Automating business processes
- Processing and analyzing data
- Creating API integration workflows
- Experimenting with multi-agent systems

The visual interface makes AI development accessible while maintaining the flexibility needed for complex production applications.

Support & Troubleshooting

If you encounter issues not covered in this guide:

1. Check the official documentation: https://docs.sim.ai
2. Review GitHub issues: https://github.com/simstudioai/sim/issues
3. Join community discussions on Hacker News
4. Contact AIGF Cohort 5 members for peer support

Appendix: Complete Installation Script

For a fresh installation, run these commands in sequence:

```
# 1. Clone repository
git clone https://github.com/simstudioai/sim.git
cd sim

# 2. Start Sim with CPU profile
docker compose -f docker-compose.ollama.yml --profile cpu up -d

# 3. Wait for services to start (2-3 minutes)
docker compose -f docker-compose.ollama.yml logs -f

# 4. Download starter model
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama pull
gemma2:2b

# 5. Verify installation
docker compose -f docker-compose.ollama.yml ps
docker compose -f docker-compose.ollama.yml exec ollama-cpu ollama list

# 6. Access application
# Open browser to http://localhost:3000
```

Document Version: 1.0
Last Updated: October 2, 2025
Author: Terris Foley, AIGF Cohort 5