i want it to be dockerized. And you're not giving everything in md format

Here's the complete project specification in proper markdown format:

```
# OpenClaw Docker Installation on MacBook Air M1 8GB - Project Specifications

## Executive Summary

This document provides comprehensive specifications for installing and running OpenClaw i

## Background & Architecture

### What is OpenClaw?
- **Type**: Open-source AI agent runtime (Node.js application)
- **Purpose**: Acts as a messaging gateway and tool execution environment that connects L
- **Not an LLM**: OpenClaw requires an external LLM provider (Ollama, Anthropic, OpenAI,

### Why Dockerize OpenClaw?
- **Security isolation**: Prevents direct system access by the agent
- **Clean environment**: Separates OpenClaw dependencies from host system
- **Easy management**: Simple container lifecycle management
- **Workspace isolation**: Config and workspace stored in mounted volumes

### Critical Constraint: Ollama Must Run Natively on macOS

**IMPORTANT**: Docker on macOS **does not** expose the Apple GPU to containers. Running C
- 50% slower performance (CPU-only inference)
- No GPU acceleration
- Significantly higher memory usage
- Thermal throttling

**Solution**: Run Ollama natively on macOS (using the GPU) and have Dockerized OpenClaw c

**References**: [web:75], [web:81], [web:78]

---

## System Requirements

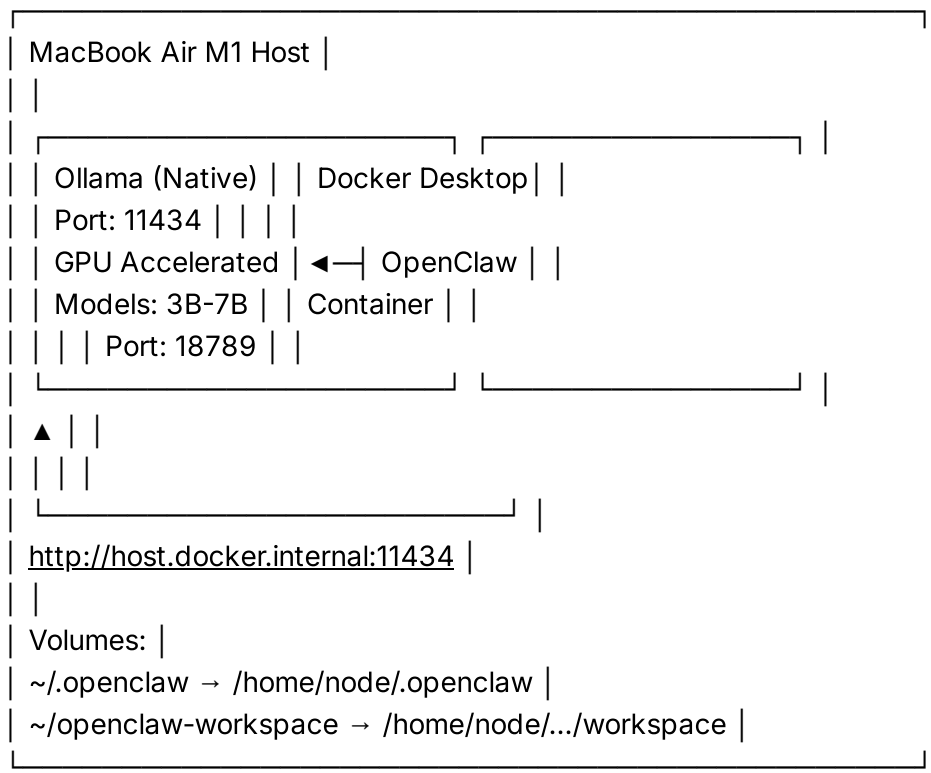
### Hardware
- MacBook Air M1 with 8GB RAM
- At least 10GB free disk space
- Active internet connection

### Software Prerequisites
- macOS 11 Big Sur or later
```

```
- Docker Desktop for Mac (Apple Silicon version)
- Terminal access
- Admin/sudo privileges

---

## Architecture Diagram
```




```
---

## Implementation Steps

### Phase 1: Install Docker Desktop

#### Step 1.1: Download and Install Docker Desktop

**Actions**:
1. Visit https://www.docker.com/products/docker-desktop/
2. Download Docker Desktop for Mac (Apple Silicon)
3. Open the downloaded DMG file
4. Drag Docker.app to Applications folder
5. Launch Docker Desktop from Applications
6. Complete the setup wizard
7. Accept the Docker service agreement
8. Wait for Docker engine to start (whale icon in menu bar should be steady)

**Verification**:
```bash
```
```

```
docker --version
docker compose version
```

Expected Output:

```
Docker version 25.x.x, build xxxxx
Docker Compose version v2.x.x
```

Troubleshooting:

- If "command not found": Ensure Docker Desktop is running (check menu bar icon)
- If Docker won't start: Check System Settings → Privacy & Security → Allow Docker

References: [web:73], [web:76]

Phase 2: Install and Configure Ollama (Native)

Step 2.1: Install Ollama on macOS

Method A: Homebrew (Recommended)

```
# Install Ollama
brew install ollama

# Start Ollama as a background service
brew services start ollama
```

Method B: Direct Download

1. Visit https://ollama.com/download
2. Download Ollama for macOS
3. Extract and move Ollama.app to Applications
4. Launch Ollama from Applications
5. Follow setup wizard to install CLI tools

Verification:

```
# Check version
ollama --version

# Verify server is running
curl http://localhost:11434/
```

Expected Output:

```
Ollama is running
```

References: [web:59], [web:62], [web:78]

Step 2.2: Download Optimal Models for 8GB RAM

Recommended Models (choose one or more):

Model	Size	Speed (tokens/sec)	Use Case	Memory
llama3.2:3b	2GB	40-82	General purpose, best quality/size	~3GB
phi3:mini	1.5GB	~45	Fastest, coding	~2.5GB
gemma:2b	1.4GB	~50	Fast, efficient	~2.5GB
qwen2.5:1.7b	1.2GB	~60	Very fast	~2GB
mistral:7b	4.1GB	~25	Coding tasks	~5GB

Installation:

```
# Download Llama 3.2 3B (recommended starting point)
ollama pull llama3.2:3b

# Or download Phi-3 Mini (fastest)
ollama pull phi3:mini

# List installed models
ollama list
```

Test Model:

```
# Run model interactively with performance stats
ollama run llama3.2:3b --verbose "Hello, what are you?"

# Exit: type /bye
```

Expected Behavior:

- Model loads in 2-5 seconds
- Response generation starts immediately
- GPU usage spikes to 85%+ (check Activity Monitor)
- Temperature stays below 80°C

References: [web:40], [web:45], [web:48]

Phase 3: Clone OpenClaw Repository

Step 3.1: Clone and Prepare

```
# Navigate to your preferred directory
cd ~/Projects

# Clone the official OpenClaw repository
git clone https://github.com/openclaw/openclaw.git

# Enter directory
cd openclaw

# List contents to verify
ls -la
```

Expected Files:

- `docker-compose.yml`
- `docker-setup.sh`
- `Dockerfile`
- `package.json`
- `.env.example`

References: [web:69], [web:79], [web:80]

Phase 4: Configure Docker Environment

Step 4.1: Create Environment Variables File

Create `.env` file:

```
# In the openclaw directory
touch .env
nano .env
```

Required Environment Variables:

```
# Image Configuration
OPENCLAW_IMAGE=openclaw:local

# Directory Configuration (adjust paths as needed)
OPENCLAW_CONFIG_DIR=${HOME}/.openclaw
OPENCLAW_WORKSPACE_DIR=${HOME}/openclaw-workspace

# Gateway Configuration
OPENCLAW_GATEWAY_TOKEN=your_secure_token_here_change_this
OPENCLAW_GATEWAY_BIND=lan
```

```
OPENCLAW_GATEWAY_PORT=18789
OPENCLAW_BRIDGE_PORT=18790

# Optional: Claude Session Keys (leave empty if not using)
CLAUDE_AI_SESSION_KEY=
CLAUDE_WEB_SESSION_KEY=
CLAUDE_WEB_COOKIE=
```

Generate Secure Token:

```
# Generate a random secure token
openssl rand -base64 32
```

Copy the output and replace `your_secure_token_here_change_this` in `.env`

Save and Exit: `Ctrl+X`, then `Y`, then `Enter`

References: [web:80], [web:79]

Step 4.2: Create Host Directories

```
# Create config directory
mkdir -p ~/.openclaw

# Create workspace directory
mkdir -p ~/openclaw-workspace

# Set proper permissions
chmod 755 ~/.openclaw
chmod 755 ~/openclaw-workspace
```

Purpose:

- `~/.openclaw`: Stores OpenClaw configuration, memory, and state
- `~/openclaw-workspace`: Agent's working directory for files

Step 4.3: Create OpenClaw Configuration File

Create config file:

```
nano ~/.openclaw/openclaw.json
```

Configuration for Docker + Native Ollama:

```
{
  "gateway": {
    "bind": "lan",
```

```
    "port": 18789,
    "auth": {
      "token": "your_secure_token_from_env_file"
    }
  },
  "llm": {
    "provider": "ollama",
    "baseURL": "http://host.docker.internal:11434/v1",
    "model": "llama3.2:3b",
    "temperature": 0.7,
    "maxTokens": 4096,
    "input": "text"
  },
  "tools": {
    "bash": {
      "enabled": true,
      "timeout": 30000
    },
    "browser": {
      "enabled": true
    },
    "file": {
      "enabled": true
    }
  },
  "channels": [],
  "memory": {
    "enabled": true,
    "type": "local"
  }
}
```

Critical Configuration Note:

- `baseURL`: Use `http://host.docker.internal:11434/v1` to allow Docker container to reach Ollama on the host Mac
- `input`: Set to `"text"` (not `"multimodal"`) to save memory on 8GB system
- `model`: Match the model name you pulled in Ollama (e.g., `llama3.2:3b`)

Alternative Models:

- Change `"model": "phi3:mini"` for fastest performance
- Change `"model": "mistral:7b"` for better coding capabilities

Save and Exit: `Ctrl+X`, then `Y`, then `Enter`

References: [web:58], [web:66], [page:2]

Phase 5: Build and Deploy Docker Container

Step 5.1: Build OpenClaw Docker Image

Option A: Use Official Setup Script (Automated)

```
# Make script executable
chmod +x docker-setup.sh

# Run the setup script
./docker-setup.sh
```

What the script does:

1. Builds the Docker image locally
2. Creates configuration directories
3. Launches interactive onboarding wizard
4. Starts the container

Follow the wizard prompts:

- **Gateway mode**: Select `lan` (accessible from your local network)
- **LLM Provider**: Select `ollama`
- **Ollama Base URL**: Enter `http://host.docker.internal:11434/v1`
- **Model name**: Enter `llama3.2:3b` (or your chosen model)
- **Channels**: Skip for now (can add later)

References: [web:73], [web:76], [web:79]

Option B: Manual Build (More Control)

```
# Build the Docker image
docker build -t openclaw:local .

# Verify image was created
docker images | grep openclaw
```

Expected Output:

```
openclaw    local    abc123def456    Just now    500MB
```

Step 5.2: Start OpenClaw Container

Using Docker Compose (Recommended):

```
# Start in detached mode (background)
docker compose up -d openclaw-gateway

# View logs in real-time
docker compose logs -f openclaw-gateway
```

Expected Log Output:

```
[OpenClaw] Starting gateway...
[OpenClaw] Binding to lan:18789
[OpenClaw] Connecting to LLM: ollama at http://host.docker.internal:11434/v1
[OpenClaw] Model: llama3.2:3b
[OpenClaw] Gateway ready at http://0.0.0.0:18789
[OpenClaw] Health check: OK
```

Troubleshooting Connection:
If you see "Cannot connect to Ollama":

```
# Test connection from inside container
docker compose exec openclaw-gateway curl http://host.docker.internal:11434/

# Should return: "Ollama is running"
```

References: [web:74], [web:80]

Step 5.3: Verify Container is Running

```
# Check container status
docker compose ps

# Should show:
# NAME                    STATUS
# openclaw-gateway        Up X minutes

# Check container health
docker compose exec openclaw-gateway node dist/index.js health
```

Phase 6: Access and Test OpenClaw

Step 6.1: Access Web Interface

Open browser and navigate to:

```
http://localhost:18789
```

Login:

- Enter the `OPENCLAW_GATEWAY_TOKEN` from your `.env` file

Expected Result: OpenClaw Control UI loads successfully

References: [web:73], [web:82]

Step 6.2: Test Basic Functionality

Test 1: Simple Query

```
You: What is 2 + 2?
```

Expected: Agent responds with "4" using the local model

Test 2: System Information

```
You: What operating system are you running on?
```

Expected: Agent uses bash tool to run `uname -a` and reports Linux (container OS)

Test 3: File Operations

```
You: Create a test file named hello.txt with "Hello World"
```

Expected: Agent creates file in workspace directory

Verify file was created:

```
ls -la ~/openclaw-workspace/
cat ~/openclaw-workspace/hello.txt
```

Step 6.3: Performance Monitoring

Monitor Resource Usage:

```
# On macOS host, check Activity Monitor:
# - Ollama should show ~3-5GB RAM usage
# - Docker Desktop should show ~1-2GB RAM usage
```

```
# - Total: 6-8GB (within 8GB limit)

# Check Docker stats
docker stats openclaw-gateway

# Watch Ollama performance
ollama ps
```

Expected Performance:

- **Response time**: 1-3 seconds for first token
- **Generation speed**: 35-80 tokens/second (depending on model)
- **Memory**: 6-8GB total system usage
- **CPU**: 20-40% average, 100% during generation bursts
- **GPU**: 85%+ during inference (check Activity Monitor → GPU)

References: [web:40], [web:45], [web:78]

Phase 7: Container Management

Step 7.1: Stop/Start/Restart

Stop container:

```
docker compose stop openclaw-gateway
```

Start container:

```
docker compose start openclaw-gateway
```

Restart container:

```
docker compose restart openclaw-gateway
```

Stop and remove:

```
docker compose down
```

Step 7.2: View Logs

Real-time logs:

```
docker compose logs -f openclaw-gateway
```

Last 100 lines:

```
docker compose logs --tail=100 openclaw-gateway
```

Save logs to file:

```
docker compose logs openclaw-gateway > openclaw-logs.txt
```

Step 7.3: Update OpenClaw

Update to latest version:

```
# Stop container
docker compose down

# Pull latest changes
git pull origin main

# Rebuild image
docker build -t openclaw:local .

# Start with new image
docker compose up -d openclaw-gateway
```

References: [web:82]

Step 7.4: CLI Management Commands

Execute CLI commands inside container:

```
# Check status
docker compose run --rm openclaw-cli status

# Check health
docker compose run --rm openclaw-cli health

# Check models
docker compose run --rm openclaw-cli models status

# Add Telegram channel
docker compose run --rm openclaw-cli channels add telegram
```

```
# View configuration
docker compose run --rm openclaw-cli config show
```

Important: All CLI commands must be run from the directory containing `docker-compose.yml`

References: [web:79]

Phase 8: Optional Enhancements

Step 8.1: Add Messaging Channels

Telegram Setup:

```
# Run CLI to add Telegram
docker compose run --rm openclaw-cli channels add telegram

# Follow prompts:
# 1. Create bot via @BotFather on Telegram
# 2. Copy bot token
# 3. Paste token when prompted
# 4. Get your Telegram user ID from @userinfobot
# 5. Enter user ID to authorize yourself
```

WhatsApp Setup:

```
# Add WhatsApp
docker compose run --rm openclaw-cli channels add whatsapp

# Scan QR code with WhatsApp mobile app
```

References: [web:58], [web:82]

Step 8.2: Configure Browser Tool (Playwright)

Enable browser automation:
The Docker container includes Playwright for browser automation. To use it:

1. Browser tool is pre-installed in the container
2. It runs headless Chromium by default
3. Configure in `openclaw.json`:

```
{
  "tools": {
    "browser": {
      "enabled": true,
      "headless": true,
      "timeout": 30000
```

```
    }
  }
}
```

Test browser tool:

```
You: Go to example.com and tell me the page title
```

References: [web:31], [web:37]

Step 8.3: Try Different Models

Switch models without rebuilding:

1. Pull new model in Ollama (on host):

```
ollama pull mistral:7b
```

2. Update `~/.openclaw/openclaw.json`:

```
{
  "llm": {
    "model": "mistral:7b"
  }
}
```

3. Restart container:

```
docker compose restart openclaw-gateway
```

Model Comparison:

- **llama3.2:3b**: Best general purpose, balanced
- **phi3:mini**: Fastest, lowest power
- **mistral:7b**: Best for coding
- **gemma:2b**: Most memory efficient

Phase 9: Security Hardening

Step 9.1: Network Isolation

Restrict to localhost only:

Edit `.env`:

```
OPENCLAW_GATEWAY_BIND=loopback
```

Restart:

```
docker compose restart openclaw-gateway
```

Now accessible only from `http://localhost:18789` (not from other devices)

Step 9.2: Strong Authentication

Use long random tokens:

```
# Generate strong token
openssl rand -base64 48

# Update .env with new token
# Update ~/.openclaw/openclaw.json with same token
```

Step 9.3: Tool Sandboxing

Limit bash tool access:

```
{
  "tools": {
    "bash": {
      "enabled": true,
      "timeout": 10000,
      "allowedCommands": ["ls", "cat", "pwd", "echo"],
      "deniedCommands": ["rm", "sudo", "curl"]
    }
  }
}
```

Disable dangerous tools:

```
{
  "tools": {
    "bash": {
      "enabled": false
    }
  }
}
```

References: [page:2], [web:79]

Step 9.4: Volume Permissions

Set read-only workspace (if needed):
Edit `docker-compose.yml`:

```
volumes:
  - ${OPENCLAW_CONFIG_DIR}:/home/node/.openclaw
  - ${OPENCLAW_WORKSPACE_DIR}:/home/node/.openclaw/workspace:ro
```

The `:ro` flag makes workspace read-only.

Troubleshooting Guide

Problem: "Cannot connect to Ollama"

Symptoms:

- Logs show: "Error connecting to http://host.docker.internal:11434"

Solutions:

1. Verify Ollama is running on host:

```
curl http://localhost:11434/
```

2. Test from inside container:

```
docker compose exec openclaw-gateway curl http://host.docker.internal:11434/
```

3. If using older Docker Desktop, try:

```
{
  "llm": {
    "baseURL": "http://docker.for.mac.host.internal:11434/v1"
  }
}
```

References: [web:75], [web:81]

Problem: Model runs very slowly

Symptoms:

- Generation speed < 10 tokens/second
- High CPU usage, low GPU usage

Causes:

- Ollama running in Docker (CPU-only)
- Wrong model quantization
- Insufficient memory

Solutions:

1. Ensure Ollama runs natively (not in Docker)
2. Verify GPU usage in Activity Monitor
3. Use smaller model (phi3:mini or gemma:2b)
4. Close other applications

References: [web:75], [web:78]

Problem: "Out of memory" errors

Symptoms:

- Container crashes with OOM
- macOS shows memory pressure warnings

Solutions:

1. Switch to smaller model:

```
ollama pull gemma:2b
```

2. Limit Docker memory:
   Open Docker Desktop → Settings → Resources → Memory Limit: 4GB
3. Set `input: "text"` (not `"multimodal"`) in config
4. Close unused applications

References: [web:45], [web:54]

Problem: Container won't start

Symptoms:

- `docker compose up` fails
- "Port already in use"

Solutions:

1. Check if port is occupied:

```
lsof -i :18789
```

2. Change port in `.env`:

```
OPENCLAW_GATEWAY_PORT=18790
```

3. Remove old containers:

```
docker compose down
docker system prune -a
```

Problem: "Permission denied" errors

Symptoms:

- Cannot write to workspace
- Cannot read config files

Solutions:

1. Fix directory permissions:

```
chmod -R 755 ~/.openclaw
chmod -R 755 ~/openclaw-workspace
```

2. Check Docker Desktop settings:
    - Settings → Resources → File Sharing
    - Ensure home directory is shared

Problem: Gateway token not working

Symptoms:

- "Invalid token" error in browser
- Cannot log in to UI

Solutions:

1. Verify tokens match in both files:

```
cat .env | grep TOKEN
cat ~/.openclaw/openclaw.json | grep token
```

2. Ensure no extra spaces or quotes
3. Restart container after changes:

```
docker compose restart openclaw-gateway
```

Performance Tuning

Memory Optimization (8GB System)

Target Distribution:

- macOS: 2GB
- Ollama + Model: 3-4GB
- Docker + OpenClaw: 1-2GB
- Browser/Apps: 1GB
- **Total: 7-8GB**

Tips:

1. Use Q4 quantized models (Ollama default)
2. Set `input: "text"` instead of `"multimodal"`
3. Limit Docker memory to 4GB max
4. Close Safari/Chrome when not needed
5. Monitor Activity Monitor → Memory tab

Speed Optimization

Fastest Configuration:

```
{
  "llm": {
    "model": "phi3:mini",
    "temperature": 0.5,
    "maxTokens": 2048
  }
}
```

Balanced Configuration:

```
{
  "llm": {
    "model": "llama3.2:3b",
    "temperature": 0.7,
    "maxTokens": 4096
  }
}
```

Quality Configuration (slower):

```
{
  "llm": {
```

```
    "model": "mistral:7b",
    "temperature": 0.8,
    "maxTokens": 8192
  }
}
```

Backup and Restore

Backup Configuration

```
# Create backup directory
mkdir -p ~/openclaw-backups

# Backup config
cp -r ~/.openclaw ~/openclaw-backups/openclaw-config-$(date +%Y%m%d)

# Backup workspace
cp -r ~/openclaw-workspace ~/openclaw-backups/openclaw-workspace-$(date +%Y%m%d)

# Backup .env
cp ~/Projects/openclaw/.env ~/openclaw-backups/.env-$(date +%Y%m%d)
```

Restore Configuration

```
# Stop container
docker compose down

# Restore config
cp -r ~/openclaw-backups/openclaw-config-20260217 ~/.openclaw

# Restore workspace
cp -r ~/openclaw-backups/openclaw-workspace-20260217 ~/openclaw-workspace

# Restart
docker compose up -d openclaw-gateway
```

Complete Command Reference

Docker Commands

```
# Build
docker build -t openclaw:local .

# Start
docker compose up -d openclaw-gateway

# Stop
```

```
docker compose stop

# Restart
docker compose restart openclaw-gateway

# Logs
docker compose logs -f openclaw-gateway

# Remove
docker compose down

# Remove with volumes
docker compose down -v

# Rebuild and restart
docker compose up -d --build openclaw-gateway
```

Ollama Commands

```
# Install
brew install ollama

# Start service
brew services start ollama

# Stop service
brew services stop ollama

# Pull model
ollama pull llama3.2:3b

# List models
ollama list

# Run model
ollama run llama3.2:3b

# Remove model
ollama rm llama3.2:3b

# Show model info
ollama show llama3.2:3b

# Check running models
ollama ps
```

OpenClaw CLI Commands (in container)

```
# Status
docker compose run --rm openclaw-cli status

# Health
docker compose run --rm openclaw-cli health
```

```
# Models
docker compose run --rm openclaw-cli models status

# Channels
docker compose run --rm openclaw-cli channels list
docker compose run --rm openclaw-cli channels add telegram
docker compose run --rm openclaw-cli channels remove telegram

# Config
docker compose run --rm openclaw-cli config show

# Logs
docker compose run --rm openclaw-cli logs

# Version
docker compose run --rm openclaw-cli --version
```

File Structure Reference

```
MacBook Air M1
├── ~/.openclaw/                    # Config directory (mounted volume)
│   ├── openclaw.json               # Main configuration
│   ├── memory/                     # Agent memory storage
│   └── logs/                       # Application logs
│
├── ~/openclaw-workspace/           # Agent workspace (mounted volume)
│   ├── files/                      # Files created by agent
│   └── downloads/                  # Downloaded content
│
└── ~/Projects/openclaw/            # Git repository
    ├── docker-compose.yml          # Docker Compose config
    ├── Dockerfile                  # Docker image definition
    ├── docker-setup.sh             # Setup script
    ├── .env                        # Environment variables
    ├── package.json                # Node.js dependencies
    └── dist/                       # Compiled application
```

Expected Outcomes

After Successful Installation

1. **Ollama runs natively** on macOS with GPU acceleration
2. **OpenClaw runs in Docker** container, isolated from host
3. **Web UI accessible** at `http://localhost:18789`
4. **Response times**: 1-3 seconds to first token
5. **Generation speed**: 35-80 tokens/second
6. **Memory usage**: 6-8GB total (within 8GB limit)

7. **GPU utilization**: 85%+ during inference
8. **CPU temperature**: 60-80°C during generation
9. **Container restarts** automatically on failure
10. **Persistent state** across container restarts

Next Steps After Installation

1. **Test all tools**: bash, browser, file operations
2. **Add messaging channels**: Telegram or WhatsApp
3. **Create workflows**: Automate repetitive tasks
4. **Install skills**: Browse https://clawhub.ai/ for extensions
5. **Configure personality**: Customize agent behavior
6. **Set up schedules**: Automated tasks via cron
7. **Monitor performance**: Check logs regularly
8. **Optimize model choice**: Balance speed vs quality
9. **Implement backups**: Regular config/workspace backups
10. **Explore integrations**: Google Workspace, Notion, etc.

Security Warnings

⚠ **Important Security Considerations**:

1. **Bash tool has full terminal access** - Be careful what you ask it to do
2. **Files in workspace are persistent** - Sensitive data persists across restarts
3. **Gateway token is the only authentication** - Keep it secret
4. **Container runs with your user permissions** - Not root, but still powerful
5. **Network access** - Agent can make outbound HTTP requests
6. **Browser tool** - Can visit websites and execute JavaScript
7. **LLM provider sees all messages** - Use local models for privacy
8. **Docker volumes persist** - Delete volumes to remove data completely

Recommended Practices:

- Never share your gateway token
- Don't run OpenClaw on public networks without VPN
- Use `bind: loopback` for local-only access
- Review bash commands before confirming execution
- Regularly audit workspace files
- Keep Docker Desktop updated

- Monitor container logs for suspicious activity

Maintenance Schedule

Daily

- Check container status: `docker compose ps`
- Monitor resource usage: Activity Monitor

Weekly

- Review logs: `docker compose logs --tail=100`
- Check for Ollama model updates
- Backup workspace directory

Monthly

- Update OpenClaw: `git pull && docker compose up -d --build`
- Update Ollama: `brew upgrade ollama`
- Update Docker Desktop: Check for updates
- Clean Docker: `docker system prune`
- Rotate gateway token (security)

Resources and References

Official Documentation

- OpenClaw Docs: https://docs.openclaw.ai/
- Ollama Docs: https://ollama.com/docs
- Docker Docs: https://docs.docker.com/

GitHub Repositories

- OpenClaw: https://github.com/openclaw/openclaw
- Docker Compose file: https://github.com/openclaw/openclaw/blob/main/docker-compose.yml

Community Resources

- ClawHub (Skills): https://clawhub.ai/
- Reddit: r/openclaw
- Discord: https://discord.gg/openclaw

Video Tutorials

- [web:71] How To Install OpenClaw With Docker
- [web:82] How To Install OpenClaw on Hostinger VPS (Docker)
- [web:56] How to: Install OpenClaw/ClawdBot on macOS 2026

Performance Benchmarks

- [web:78] LLM Performance on Mac: Native vs Docker Ollama Benchmark
- [web:40] I tested 10 LLMs locally on my MacBook Air M1 (8GB RAM)

Success Criteria

Installation is successful when:

- [ ] Ollama runs natively on macOS and responds to `curl http://localhost:11434/`
- [ ] At least one model is downloaded: `ollama list` shows model
- [ ] Docker Desktop is running: `docker --version` works
- [ ] OpenClaw container is running: `docker compose ps` shows "Up"
- [ ] Web UI loads: `http://localhost:18789` accessible
- [ ] Authentication works: Gateway token accepted
- [ ] Agent responds: Simple query returns answer within 5 seconds
- [ ] Tools work: Bash command executes successfully
- [ ] GPU is used: Activity Monitor shows GPU usage during generation
- [ ] Memory is stable: Total system memory < 8GB
- [ ] Container persists: `docker compose restart` preserves configuration

Glossary

- **OpenClaw**: Open-source AI agent runtime (Node.js application)
- **Ollama**: Local LLM runtime for macOS/Linux
- **LLM**: Large Language Model (e.g., Llama, Mistral, Phi)
- **Gateway**: OpenClaw's web server that handles requests
- **Docker Compose**: Tool for defining multi-container Docker applications

- **Volume**: Persistent storage mounted into Docker container
- **host.docker.internal**: Special DNS name for host machine from container
- **Q4 Quantization**: 4-bit quantization for smaller model size
- **Tokens/second**: Speed metric for text generation
- **Tool**: Function the LLM can call (bash, browser, file)
- **Channel**: Messaging platform integration (Telegram, WhatsApp)

End of Specification Document

Version: 1.0
Date: February 17, 2026
Target Platform: MacBook Air M1 8GB RAM
Architecture: Docker + Native Ollama

This specification is ready to be provided to Claude or any coding agent for implementation assistance.

```
This is now a complete, properly formatted Markdown document with all instructions for Do
<span style="display:none">[^1][^10][^11][^12][^13][^14][^15][^2][^3][^4][^5][^6][^7][^8]

<div align="center">⁂</div>

[^1]: https://til.simonwillison.net/llms/openclaw-docker
[^2]: https://www.youtube.com/watch?v=fcZMmP5dsl4
[^3]: https://www.youtube.com/watch?v=-aeR1cQktdM
[^4]: https://www.youtube.com/watch?v=Qx97iDCCccM
[^5]: https://www.hostinger.com/tutorials/how-to-set-up-openclaw
[^6]: https://openclawwiki.org/tools/docker-compose-generator
[^7]: https://github.com/ollama/ollama/issues/3849
[^8]: https://www.hostinger.com/uk/tutorials/how-to-set-up-openclaw
[^9]: https://openclaw.bz/blog/openclaw-docker-deploy
[^10]: https://www.vchalyi.com/blog/2025/ollama-performance-benchmark-macos/
[^11]: https://aimlapi.com/blog/running-openclaw-in-docker-secure-local-setup-and-practic
[^12]: https://github.com/openclaw/openclaw/blob/main/docker-compose.yml
[^13]: https://chariotsolutions.com/blog/post/apple-silicon-gpus-docker-and-ollama-pick-t
[^14]: https://www.youtube.com/watch?v=XvEDmYObHaI
[^15]: https://www.openclawinstall.info/openclaw-install-docker/
```