

Vigthoria CLI User Handbook

AI-Powered Terminal Coding Assistant

Version 2.0.0

© 2026 Vigthoria Technologies

Table of Contents

1. Introduction
 2. CLI Visual Overview
 3. Installation
 4. Quick Start Guide
 5. Authentication
 6. Interactive Chat Mode
 7. Agent Mode (Autonomous)
 8. Session Management
 9. File Operations
 10. Code Generation
 11. Code Review & Fixes
 12. Configuration
 13. Available AI Models
 14. Local Mode with Vigthoria Inference
 15. Distributed Compute (Coming Soon)
 16. Integration with Vigthoria Coder
 17. Vigthoria Repository
 18. Vigthoria Hub (API Marketplace)
 19. Deploy & Host
 20. Sophie — Autonomous Loop Engine
 21. Squad — Multi-Agent Team System
 22. Command Reference
 23. Workflows & Examples
 24. Troubleshooting
 25. FAQ
-

Introduction

What is Vigthoria CLI?

Vigthoria CLI is a powerful terminal-based coding assistant that brings the full capabilities of Vigthoria's AI models to your command line. Whether you're working locally, over SSH, or in a headless environment, Vigthoria CLI provides intelligent code assistance without leaving your terminal.

Key Features

Feature	Description
Interactive Chat	Natural language coding assistance with context awareness

Feature	Description
<ul style="list-style-type: none"> ☐ Agent Mode Session Management File Editing Code Generation Code Explanation Bug Fixing Code Review Multi-Model Support Project Context ☐ Local Mode ⊗ Sophie Loop Engine ☐ Squad Multi-Agent 	<p>Autonomous autonomous file editing and command execution</p> <p>Save, resume, and manage conversation sessions</p> <p>AI-powered code modifications with diff preview</p> <p>Generate complete code from descriptions</p> <p>Understand complex code with detailed explanations</p> <p>Automatic issue detection and resolution</p> <p>Quality analysis with actionable suggestions</p> <p>Access to all Vigthoria AI models based on subscription</p> <p>Understands your project structure and dependencies</p> <p>Works offline with Ollama for local development</p> <p>Autonomous AI iteration loops — give a task, Sophie loops until done</p> <p>Deploy teams of specialized AI agents to build entire projects</p>

Who Should Use This?

- **Backend Developers** - Working primarily in terminal environments
- **DevOps Engineers** - Scripting and automation tasks
- **Remote Workers** - SSH sessions to remote servers
- **Power Users** - Prefer keyboard-driven workflows
- **CI/CD Integration** - Automated code generation and review

Requirements

Component	Minimum	Recommended
Node.js	18.0.0	20.x LTS
npm	8.x	10.x
OS	Linux, macOS, Windows	Linux/macOS
Internet	Required	Stable connection
Terminal	Any	iTerm2, Warp, Kitty

CLI Visual Overview

Main CLI Banner

When you run `vig --version` or start any command, you'll see:

```

VIGTHORIA CLI - AI-Powered Coding Assistant
Version 1.4.1

```

Command Overview

```
$ vig --help
```

Usage: vigthoria [options] [command]

AI-powered terminal coding assistant for Vigthoria Coder subscribers

Options:

-V, --version	output the version number
-h, --help	display help for command

Commands:

chat c [options]	Start interactive chat with Vigthoria AI
agent a [options]	Start agentic mode - AI can read/write files
edit e [options] <file>	Edit a file with AI assistance
generate g [options] <description>	Generate code from description
explain x [options] <file>	Explain code in a file
fix f [options] <file>	Fix issues in a file
review r [options] <file>	Review code quality
hub marketplace	Discover and activate Vigthoria API modules
repo repository	Push and pull projects to/from Vigthoria Repo
deploy host [options]	Deploy and host your project
login [options]	Login to Vigthoria Coder
logout	Logout from Vigthoria Coder
status	Show authentication and subscription status
config [options]	Configure Vigthoria CLI settings
update upgrade [options]	Check for updates and upgrade
init	Initialize Vigthoria in current project

Chat Session Example

```
$ vig chat
```

VIGTHORIA CLI - AI-Powered Coding Assistant
Version 1.4.1

```
[Chat Mode] Model: vigthoria-code | Session: new
```

```
you > What is VWF?
```

```
vigthoria > VWF (Vigthoria Web Framework) is a modern web development framework created by Vigthoria Technologies. It features:
```

- Server-side rendering
- Built-in authentication
- Database integration
- RESTful API generation

```
you > /help
```

Available Commands:

```
/file <path> - Add file to context
```

```
/edit <path> - Switch to edit mode
/diff        - Show pending changes
/apply       - Apply pending changes
/model <name> - Switch AI model
/agent       - Toggle agent mode
/clear       - Clear conversation
/sessions    - List sessions
/save        - Save session
/exit        - Exit (auto-saves)
```

```
you > /exit
Session saved. Goodbye!
```

Agent Mode Example

```
$ vig agent --auto-approve
```

VIGTHORIA CLI - AI-Powered Coding Assistant
Version 1.4.1

```
[Agent Mode] Model: vighthoria-code | Auto-approve: ON
```

```
you > Create a REST API for user management
```

```
vighthoria > I'll create a complete REST API. Let me analyze and implement...
```

```
⊛ Executing: list_dir
  Path: .
✓ Found: package.json, src/

⊛ Executing: write_file
  File: src/api/users.ts
✓ Created user API routes

⊛ Executing: write_file
  File: src/models/user.ts
✓ Created user model

⊛ Executing: bash
  Command: npm install express mongoose bcrypt
✓ Dependencies installed
```

```
Done! Created:
• src/api/users.ts - CRUD endpoints
• src/models/user.ts - User schema
• Updated package.json
```

Repository Commands

```
$ vig repo --help
```

```
Usage: vighthoria repo|repository [options] [command]
```


- Code Assistance
- Text & Content

Selected: Music & Audio Generation

Available Modules:

Module	Description	Pricing
vigthoria-music-ai	Full music generation suite	\$0.02/gen
ace-step-duet	Duet & collaboration AI	\$0.03/gen
audio-enhance	Audio quality enhancement	\$0.01/file

Deploy Commands

```
$ vig deploy --help
```

Usage: vigthoria deploy|host [options] [command]

Deploy and host your project on Vigthoria infrastructure

Commands:

```

preview [options]           Deploy to free preview URL
subdomain [options] <name> Deploy to name.vigthoria.io
custom [options] <domain>  Deploy to custom domain
list|ls                    List deployments
plans                      Show hosting plans
status [domain]           Check status
verify <domain>          Verify DNS
remove|rm <domain>       Remove deployment

```

```
$ vig deploy preview
```

Vigthoria Deploy - Preview

- Building project...
- Uploading to Vigthoria Cloud...
- Deploying...

Deployed successfully!

- Preview URL: <https://preview-x7k2m.vigthoria.io>
- Expires: 7 days
- Upgrade to subdomain for permanent URL

Installation

Quick Install (Recommended)

```
curl -fsSL https://cli.vigthoria.io/install.sh | bash
```

This script will: 1. Check Node.js version 2. Install Vigthoria CLI globally 3. Set up shell completions 4. Create command shortcuts

npm Global Install

```
npm install -g vigthoria-cli
```

Manual Installation

Clone from Vigthoria Market

```
git clone https://market.vigthoria.io/vigthoria/vigthoria-cli.git
cd vigthoria-cli
```

Or from Vigthoria Community

```
git clone https://community.vigthoria.io/repos/vigthoria-cli.git
cd vigthoria-cli
```

Install dependencies

```
npm install
```

Build

```
npm run build
```

Link globally

```
npm link
```

Verify Installation

```
vigthoria --version
```

Output: 1.0.0

```
vigthoria --help
```

Shows all available commands

Update

```
npm update -g vigthoria-cli
```

Quick Start Guide

Step 1: Login

```
vigthoria login
```

Choose your login method: - **Email & Password** - Enter your Vigthoria account credentials - **API Token** - Use a token from your Vigthoria Coder dashboard - **Browser Login** - OAuth flow (opens browser)

Step 2: Check Status

```
vigthoria status
```

This shows: - Account information - Subscription plan - Available models - API connectivity

Step 3: Start Coding

```
# Start interactive chat
vigthoria chat
```

```
# Or use the short alias
vig c
```

Step 4: Your First AI Conversation

you > Create a function to validate email addresses in TypeScript

vigthoria >

Here's a robust email validation function:

```
```typescript
function validateEmail(email: string): boolean {
 const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
 return emailRegex.test(email);
}

// Enhanced version with detailed validation
interface EmailValidationResult {
 valid: boolean;
 reason?: string;
}

function validateEmailDetailed(email: string): EmailValidationResult {
 if (!email) {
 return { valid: false, reason: 'Email is required' };
 }

 if (email.length > 254) {
 return { valid: false, reason: 'Email too long' };
 }

 const regex = /^[a-zA-Z0-9.!#$%&'*/+=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)*$/;

 if (!regex.test(email)) {
 return { valid: false, reason: 'Invalid email format' };
 }

 return { valid: true };
}

Authentication
Login Methods
Email & Password
```bash
```

```
vigthoria login
# Select: Email & Password
# Enter your email and password
```

API Token Get your token from: <https://coder.vigthoria.io/settings> (API Keys section)

Interactive

```
vigthoria login
# Select: API Token
# Paste your token
```

Or directly

```
vigthoria login --token YOUR_API_TOKEN
```

Environment Variable For CI/CD or automated environments:

```
export VIGTHORIA_TOKEN="your-api-token"
vigthoria status # Will use the environment token
```

Check Authentication Status

```
vigthoria status
```

Output:

== Account Status ==

Account:

```
Email: user@example.com
User ID: usr_abc123
```

Subscription:

```
Plan: PRO
Status: Active
Expires: 2026-12-31 (365 days)
```

Available Models:

- vigthoria-fast - Quick responses (1.1B)
- vigthoria-mini - Balanced (3.8B)
- vigthoria-code - Code specialist (8B)
- vigthoria-creative - Creative (9B)

API Status:

```
Server: Online
Endpoint: https://coder.vigthoria.io
```

Logout

```
vigthoria logout
```

Interactive Chat Mode

The chat mode is the primary way to interact with Vigthoria AI. It maintains conversation context and understands your project.

Starting Chat

```
vigthoria chat
```

```
# or
```

```
vig c
```

```
# With specific model
```

```
vigthoria chat --model pro
```

```
# With project context
```

```
vigthoria chat --project /path/to/project
```

Chat Commands

While in chat mode, use these special commands:

Command	Description
/file <path>	Add a file's content to the conversation
/edit <path>	Start editing a specific file
/diff	Show pending changes
/apply	Apply pending changes to files
/model <name>	Switch AI model
/clear	Clear conversation history
/help	Show available commands
/exit or /quit	Exit chat mode

Example Chat Session

```
VIGTHORIA CLI - AI-Powered Coding Assistant
```

```
Vigthoria Chat  
Model: vigthoria-code  
Plan: PRO  
Project: /home/user/my-project
```

Type your message or /help for commands. Press Ctrl+C to exit.

```
you > /file src/api.ts
```

```
✓ Added src/api.ts to context (245 lines)
```

```
you > Add rate limiting to the authentication endpoint
```

```
vigthoria >
```

```
I'll add rate limiting to your authentication endpoint. Based on the code in
```

src/api.ts, here's what I recommend:

[... detailed response with code ...]

```
you > /edit src/api.ts
What changes would you like to make?
> Add the rate limiting code you just showed
```

```
[Diff preview]
- export async function login(req, res) {
+ import rateLimit from 'express-rate-limit';
+
+ const authLimiter = rateLimit({
+   windowMs: 15 * 60 * 1000,
+   max: 5
+ });
+
+ export async function login(req, res) {
```

```
Apply changes? (Y/n) y
✓ Changes applied to src/api.ts
```

Context Management

The chat remembers your conversation and any files you've added. To start fresh:

```
you > /clear
✓ Conversation cleared
```

□ Agent Mode

Agent mode is Vigthoria CLI's most powerful feature - Vigthoria's most advanced AI capability. It enables the AI to autonomously:

- □ **Read files** - Examine any file in your project
- ☞ **Write files** - Create or modify files
- □ **Search** - Grep patterns and glob files
- □ **Execute** - Run shell commands
- □ **Browse** - List directories
- □ **Git** - Run git commands

Starting Agent Mode

```
# Direct agent command
vigthoria agent
```

```
# Or with shortcut
vig a
```

```
# Or enable in chat
vigthoria chat --agent
vigthoria chat -a
```

Agent Options

Option	Description	Default
-m, --model <model>	AI model to use	balanced
-p, --project <path>	Project context path	Current directory
-l, --local	Use local Ollama	false
--auto-approve	Auto-approve all actions	false
-r, --resume	Resume last session	false

Available Tools

The AI can use these tools when in agent mode:

Tool	Description	Example Use
read_file	Read file contents with line range	"Read lines 1-50 of index.ts"
write_file	Create or overwrite a file	"Create a new config.ts file"
edit_file	Search & replace in file	"Change the function name"
bash	Execute shell commands	"Run npm install express"
grep	Search pattern in files	"Find all TODO comments"
list_dir	List directory contents	"Show files in src/"
glob	Find files by pattern	"Find all *.test.ts files"
git	Run git commands	"git status"

Permission System

By default, agent mode **asks for permission** before: - Writing or editing files - Running shell commands - Making destructive changes

```
△ The AI wants to use write_file
  File: src/utils/helper.ts
  Content: [Preview shown]
```

```
Allow this action? [y/N] y
✓ write_file: Success
```

Auto-Approve Mode

△ **WARNING: DANGEROUS** - Only use in trusted environments!

Auto-approve mode allows all AI actions without confirmation:

```
vigthoria agent --auto-approve
```

Or toggle within a session:

```
you > /approve
✓ Auto-approve: ON - All actions will execute automatically
```

Example Agent Session

```
VIGTHORIA CLI - AI-Powered Coding Assistant
Version 1.4.1
```

[Agent Mode] Model: vighthoria-code | Auto-approve: OFF

you > Create an Express server with TypeScript and add user authentication

vighthoria > I'll create an Express server with authentication for you. Let me:

1. Check your current project structure
2. Create the necessary files
3. Set up dependencies

⊗ Executing: list_dir

Path: .

✓ list_dir: package.json, src/, tsconfig.json

⊗ Executing: read_file

File: package.json, Lines: 1-30

✓ read_file: Success (25 lines)

⊗ Executing: write_file

File: src/server.ts

Creating Express server with authentication...

Allow? [y/N] y

✓ write_file: Success

⊗ Executing: write_file

File: src/middleware/auth.ts

Allow? [y/N] y

✓ write_file: Success

⊗ Executing: bash

Command: npm install express jsonwebtoken bcryptjs @types/express @types/jsonwebtoken

Allow? [y/N] y

✓ bash: Success

I've created your Express server with JWT authentication!

Files created:

- src/server.ts - Main server file
- src/middleware/auth.ts - Authentication middleware
- src/routes/auth.ts - Login/register routes

Run `npm run dev` to start the server.

Agent vs Chat Mode Comparison

Feature	Chat Mode	Agent Mode
Conversation	☐	☐
File Context	Manual (/file)	Automatic
Read Files	☐	☐
Write Files	Via /edit	Direct
Run Commands	☐	☐

Feature	Chat Mode	Agent Mode
Autonomous Actions Best For	<input type="checkbox"/> Q&A, discussion	<input type="checkbox"/> Implementation

Session Management

Sessions allow you to **save and resume** conversations, preserving context across multiple work sessions.

How Sessions Work

- Each chat/agent session has a unique ID
- Sessions are saved in `~/.vigthoria/sessions/`
- Sessions are project-specific (based on working directory)
- Auto-saved when you exit with `/exit`

Viewing Sessions

```
you > /sessions
```

```
== Saved Sessions ==
```

```
m1abc123-xyz [agent] (current)
  /home/user/my-project - 1/13/2026, 8:30:00 PM
m1def456-uvw
  /home/user/other-project - 1/12/2026, 3:15:00 PM
m1ghi789-rst [agent]
  /home/user/my-project - 1/11/2026, 2:00:00 PM
```

Resuming a Session

```
# Resume the last session for current project
```

```
vigthoria chat --resume
vigthoria chat -r
```

```
# Or in agent mode
```

```
vigthoria agent --resume
```

Saving Sessions

Sessions are saved automatically on `/exit`. Force save manually:

```
you > /save
✓ Session saved: m1abc123-xyz
```

Starting a New Session

```
you > /new
✓ Previous session saved
✓ New session: m1jkl012-mno
```

Viewing Conversation History

```
you > /history
```

```
== Conversation History ==
```

1. you: How do I create a REST API...
2. vighthoria: Here's how to create a REST API...
3. you: Can you add authentication...
4. vighthoria: I'll add JWT authentication...

Compacting Long Sessions

When conversations get long, compress old messages:

```
you > /compact
```

```
✓ Compacted 12 messages into summary
```

This summarizes older messages to reduce context size while preserving important information.

Session File Location

```
~/.vighthoria/sessions/  
├─ mlabc123-xyz.json  
├─ mldef456-uvw.json  
└─ mlghi789-rst.json
```

File Operations

Edit a File

```
vighthoria edit <file> [options]
```

Options: `--i`, `--instruction <text>` - Editing instruction - `-m`, `--model <model>` - AI model to use

Examples:

```
# Interactive editing
```

```
vighthoria edit src/utils.ts
```

```
# With instruction
```

```
vighthoria edit src/api.ts -i "Add error handling to all async functions"
```

```
# With specific model
```

```
vighthoria edit src/complex.ts -m pro
```

Edit Workflow

1. File content is loaded
2. You describe the changes
3. AI generates the modified code
4. Diff is shown for review
5. You approve or reject changes
6. Backup is created automatically

Fix Code Issues

vigthoria fix <file> [options]

Options: - -t, --type <type> - Fix type: bugs, style, security, performance - --apply - Auto-apply fixes without confirmation

Examples:

Find and fix bugs

vigthoria fix src/index.ts -t bugs

Fix security issues with auto-apply

vigthoria fix src/auth.ts -t security --apply

Fix performance issues

vigthoria fix src/heavy-computation.ts -t performance

Explain Code

vigthoria explain <file> [options]

Options: - -l, --lines <range> - Line range (e.g., 1-50) - -d, --detail <level> - brief, normal, detailed

Examples:

Explain entire file

vigthoria explain src/algorithm.ts

Explain specific lines

vigthoria explain src/utils.ts -l 50-100

Detailed explanation

vigthoria explain src/complex.ts -d detailed

Code Generation

Generate Code from Description

vigthoria generate <description> [options]

Options: - -l, --language <lang> - Target language (default: typescript) - -o, --output <file> - Output file path - -m, --model <model> - AI model to use

Supported Languages

- TypeScript / JavaScript
- Python
- Rust
- Go
- Java
- C# / C++
- Ruby
- PHP
- Swift

- Kotlin

Examples

```
# Generate TypeScript code
```

```
vigthoria generate "REST API endpoint for user authentication"
```

```
# Generate Python with output file
```

```
vigthoria generate "Database migration script for PostgreSQL" -l python -o migrate.py
```

```
# Generate with specific model
```

```
vigthoria generate "High-performance caching layer" -l rust -m pro
```

Generated Code Quality

Vigthoria generates production-ready code with: - Proper type definitions - Error handling - Input validation - Documentation comments - Modern best practices

Code Review & Fixes

Code Review

```
vigthoria review <file> [options]
```

Options: - -f, --format <format> - text, json, markdown

Review Output

```
== Reviewing: src/api.ts ==
```

```
Language: typescript | Lines: 245
```

```
Quality Score: 78/100
```



```
== Issues (4) ==
```

```
x [security] Line 45: Potential SQL injection vulnerability
```

```
△ [performance] Line 112: Unnecessary re-render in loop
```

```
△ [style] Line 67: Magic number should be a named constant
```

```
□ [info] Line 189: Consider using optional chaining
```

```
== Suggestions ==
```

1. Add input validation to the createUser function
2. Consider implementing rate limiting on public endpoints
3. Use environment variables for configuration values
4. Add JSDoc comments to exported functions

Export Review to File

```
# Markdown report
```

```
vigthoria review src/app.ts -f markdown > code-review.md
```

```
# JSON for CI/CD
vigthoria review src/app.ts -f json > review-results.json
```

Configuration

Interactive Configuration

```
vigthoria config
```

This opens an interactive menu to configure: - Default AI model - Color theme - Auto-apply fixes - Show diffs - Max tokens

Set Individual Values

```
vigthoria config --set model=vigthoria-code
vigthoria config --set theme=dark
vigthoria config --set autoApply=true
vigthoria config --set maxTokens=8192
```

Get Values

```
vigthoria config --get model
# Output: vigthoria-code
```

List All Settings

```
vigthoria config --list
```

Reset to Defaults

```
vigthoria config --reset
```

Initialize Project

Create a project-specific configuration:

```
cd /path/to/project
vigthoria init
```

This creates `.vigthoria.json` with project settings.

Configuration File Location

- **Global:** `~/.config/vigthoria-cli/config.json`
- **Project:** `./.vigthoria.json`

Project settings override global settings.

Available AI Models


```

└─ Load balancing and fallback
└─ API: http://localhost:4009/api/vigthoria/chat
    │
    ▼ (if unavailable)

Strategy 3: Ollama (Port 11434)
└─ Generic open-source models
└─ Fallback for development
└─ API: http://localhost:11434/api/generate

```

Prerequisites

Install Vigthoria Inference from Vigthoria Market:

```

# Clone from Vigthoria Market
git clone https://market.vigthoria.io/vigthoria/vigthoria-inference.git
cd vigthoria-inference

# Or from Vigthoria Community
git clone https://community.vigthoria.io/repos/vigthoria-inference.git
cd vigthoria-inference

# Install dependencies
pip install -r requirements.txt

# Start Vigthoria Inference Server (Port 8010)
python3 vigthoria_inference_server_v2.py

```

OpenAI-Compatible API

Vigthoria Inference Server exposes an **OpenAI-compatible API**:

```

# Direct API call to Vigthoria Inference
curl http://localhost:8010/v1/chat/completions \
  -H "Content-Type: application/json" \
  -d '{
    "model": "vigthoria-v2-code-8b",
    "messages": [{"role": "user", "content": "Write hello world in Python"}],
    "max_tokens": 512,
    "temperature": 0.7
  }'

# Response format (OpenAI compatible)
{
  "id": "vigthoria-123",
  "model": "vigthoria-v2-code-8b",
  "choices": [{
    "message": {
      "role": "assistant",
      "content": "print('Hello, World!')"
    }
  }],
  "usage": {

```

```

    "prompt_tokens": 15,
    "completion_tokens": 8,
    "total_tokens": 23
  }
}

```

Available Native Vigthoria Models

Vigthoria Inference provides access to all native Vigthoria LLMs:

```

# Mini (0.6B) - Ultra-fast responses
vigthoria-mini-0.6b

```

```

# Fast (1.7B) - Quick streaming
vigthoria-fast-1.7b

```

```

# Balanced (4B) - General purpose
vigthoria-balanced-4b

```

```

# Code (8B) - Code specialist
vigthoria-code-v2-8b

```

```

# Creative (9B) - Creative writing
vigthoria-creative-9b-v4

```

```

# Music (4B) - Music production
vigthoria-music-master-4b

```

Using Local Mode

```

# Start chat with Vigthoria Inference
vigthoria chat --local
vigthoria chat -l

```

```

# Agent mode with local inference
vigthoria agent --local

```

```

# With specific Vigthoria model
vigthoria chat --local --model code
vigthoria chat -l -m balanced

```

Local Model Mapping

When using `--local`, Vigthoria CLI connects to Vigthoria Inference (Port 8010):

Model Alias	Vigthoria Model	Size
mini	vigthoria-mini-0.6b	0.6B
fast	vigthoria-fast-1.7b	1.7B
balanced	vigthoria-balanced-4b	4B
code	vigthoria-code-v2-8b	8B
creative	vigthoria-creative-9b-v4	9B
music	vigthoria-music-master-4b	4B

Advantages of Local Mode

Benefit	Description
Privacy	Code never leaves your machine
Native Models	Full power of Vigthoria LLMs
Offline	Works without internet
No Auth	No login required
Speed	Low latency for local inference
Free	No API costs

System Requirements

Model	VRAM Required	Recommended GPU
Mini (0.6B)	1 GB	Any GPU
Fast (1.7B)	2 GB	GTX 1060+
Balanced (4B)	4 GB	RTX 3060+
Code (8B)	8 GB	RTX 3070+
Creative (9B)	10 GB	RTX 4080+

Note: With 4-bit quantization, VRAM requirements are reduced by ~40%.

Example Local Session

```
$ vigthoria chat -l -m code
```

```
VIGTHORIA CLI - AI-Powered Coding Assistant  
Version 1.4.1
```

```
[Local Mode] Model: vigthoria-code-v2-8b | Inference: localhost:8010
```

```
you > Write a Python function to reverse a string
```

```
vigthoria > Here's a Python function to reverse a string:
```

```
def reverse_string(s: str) -> str:  
    """Reverse the input string."""  
    return s[::-1]  
  
# Alternative using reversed()  
def reverse_string_alt(s: str) -> str:  
    return ''.join(reversed(s))
```

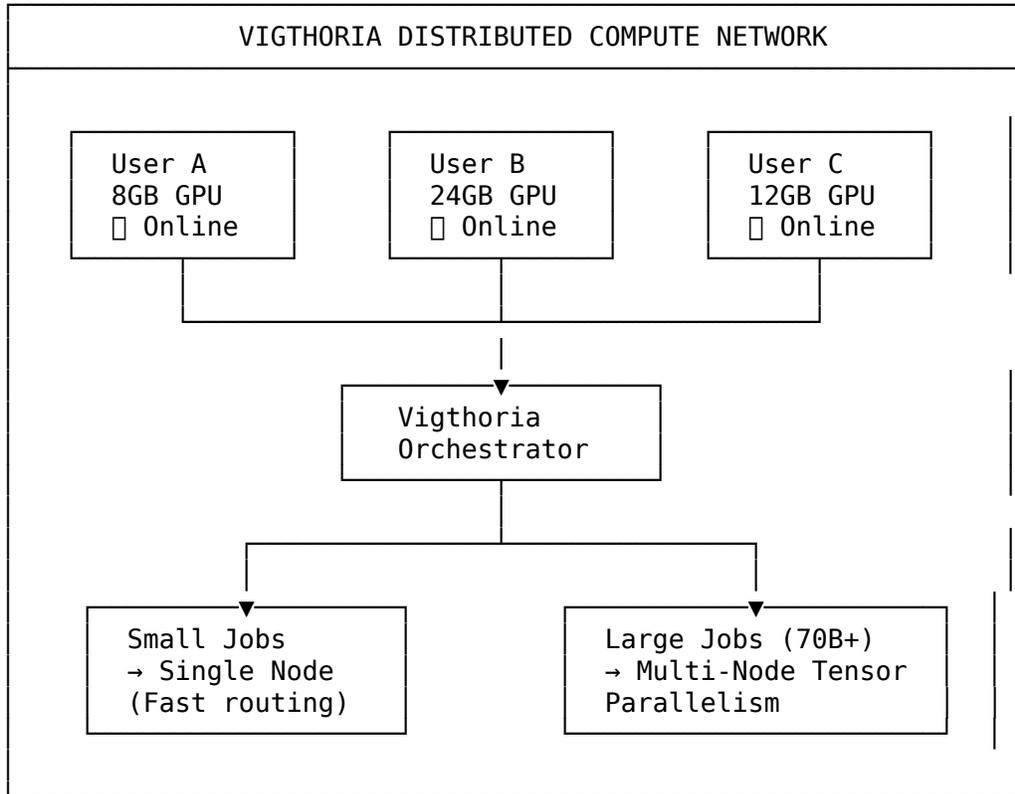
📦 Distributed Compute (Coming Soon)

Vision: User-Powered AI

Vigthoria is building a distributed compute network that allows users to: 1. **Contribute GPU power** - Share your local GPU with the Vigthoria network 2. **Earn rewards** - Receive Vigthoria

Credits for compute contributions 3. **Access larger models** - Run 32B, 70B, and 128B models across multiple GPUs

How It Works



Commands (Coming Soon)

Enable compute sharing
vigthoria compute enable

Check contribution stats
vigthoria compute status

Configure compute limits
vigthoria compute config --max-vram 80%

View earned credits
vigthoria credits

Reward System

Action	Credits Earned
1 hour compute contribution	10 Credits
Process small inference job	1 Credit
Process large inference job	5 Credits

Action	Credits Earned
Participate in distributed 70B	20 Credits

Use Credits For

- Free access to premium models
 - Extended context windows (32K, 64K, 128K)
 - Priority queue for inference
 - Storage on Vigthoria Market
-

Integration with Vigthoria Coder

Shared Authentication

Vigthoria CLI uses the same authentication as: - coder.vigthoria.io (Web IDE) - vsuser.vigthoria.io (VS Code IDE) - extension.vigthoria.io (VS Code Extension)

One account works everywhere.

Accessing Cloud Projects

Your projects from Vigthoria Coder are accessible:

```
# List cloud projects (coming soon)
```

```
vigthoria projects list
```

```
# Clone a cloud project (coming soon)
```

```
vigthoria projects clone <project-id>
```

Syncing with Web IDE

Changes made via CLI can be synced with your cloud workspace:

```
# Push local changes (coming soon)
```

```
vigthoria sync push
```

```
# Pull cloud changes (coming soon)
```

```
vigthoria sync pull
```

Subscription Benefits

Your subscription plan determines: - Available AI models - Token limits per request - Priority processing - Advanced features

📁 Vigthoria Repository

Your personal cloud repository for storing and sharing projects. Push your code to Vigthoria's secure cloud storage and access it from anywhere.

Push a Project

Push current directory

```
vigthoria repo push
```

Push specific path

```
vigthoria repo push /path/to/project
```

Push with visibility setting

```
vigthoria repo push --visibility private # Only you can see
```

```
vigthoria repo push --visibility public # Anyone can view
```

```
vigthoria repo push --visibility restricted # Only with link
```

Force overwrite existing

```
vigthoria repo push --force
```

Pull a Project

Pull a project from your repo

```
vigthoria repo pull my-project
```

Pull to specific directory

```
vigthoria repo pull my-project --output /path/to/destination
```

Overwrite existing directory

```
vigthoria repo pull my-project --force
```

List Your Projects

List all projects

```
vigthoria repo list
```

or

```
vigthoria repo ls
```

Filter by visibility

```
vigthoria repo list --visibility private
```

Check Sync Status

Check current project sync status

```
vigthoria repo status
```

Share Projects

Generate shareable link (default 7 days)

```
vigthoria repo share my-project
```

Custom expiration

```
vigthoria repo share my-project --expires 24h
```

```
vigthoria repo share my-project --expires 30d
```

Delete a Project

```
# Remove from cloud (local files unchanged)
vigthoria repo delete my-project
# or
vigthoria repo rm my-project
```

Clone Public Projects

```
# Clone from preview URL
vigthoria repo clone https://coder.vigthoria.io/preview/123
```

```
# Clone by project ID
vigthoria repo clone 123
```

📦 Vigthoria Hub (API Marketplace)

Discover and activate powerful AI APIs from Vigthoria's module marketplace. Pay-as-you-go billing for music generation, video AI, voice cloning, and more.

Discover Modules

```
# Interactive discovery wizard
vigthoria hub discover
# or
vig hub d
```

The discovery wizard will ask about your project needs and recommend the best modules.

Search Modules

```
# Semantic search for modules
vigthoria hub search "generate background music for my app"
vigthoria hub search "create AI voice narration"
vigthoria hub search "video scene detection"
```

List All Modules

```
# Show all available modules
vigthoria hub list
# or
vigthoria hub ls

# Filter by category
vigthoria hub list --category music
vigthoria hub list --category video
vigthoria hub list --category voice
```

View Module Details

```
# Get detailed info about a module
vigthoria hub info vigthoria-music-ai
vigthoria hub details vigthoria-voice-clone
```

Activate a Module

```
# Activate for your API key (enables pay-as-you-go)
vigthoria hub activate vigthoria-music-ai
vigthoria hub enable vigthoria-voice-clone
```

View Active Modules

```
# Show your currently active modules
vigthoria hub active
```

Available Module Categories

Category	Example Modules	Description
Music	vigthoria-music-ai, ace-step-duet	AI music generation
Voice	vigthoria-voice-clone, tts-multilang	Voice cloning & TTS
Video	vigthoria-video-gen, scene-detect	Video AI processing
Image	flux-image-gen, style-transfer	Image generation
Code	vigthoria-coder-api, code-review	Code assistance API

□ Deploy & Host

Deploy your projects to Vigthoria's hosting infrastructure with one command.

Preview Deployment

```
# Deploy to free preview URL
vigthoria deploy preview
```

You'll get a URL like: <https://preview-abc123.vigthoria.io>

Subdomain Deployment

```
# Deploy to yourname.vigthoria.io
vigthoria deploy subdomain myproject
```

```
# Result: https://myproject.vigthoria.io
```

Custom Domain

```
# Deploy to your own domain
vigthoria deploy custom myapp.com
```

```
# Verify DNS configuration
vigthoria deploy verify myapp.com
```

Manage Deployments

List all deployments

```
vigthoria deploy list
```

Check deployment status

```
vigthoria deploy status myproject.vigthoria.io
```

Remove a deployment

```
vigthoria deploy remove myproject.vigthoria.io
```

View Hosting Plans

Show available hosting plans and pricing

```
vigthoria deploy plans
```

Command Reference

Core Commands

Command	Alias	Description
vigthoria chat	vig c	Start interactive chat
vigthoria agent	vig a	Start agent mode (Autonomous)
vigthoria edit <file>	vig e	Edit file with AI
vigthoria generate <desc>	vig g	Generate code
vigthoria explain <file>	vig x	Explain code
vigthoria fix <file>	vig f	Fix code issues
vigthoria review <file>	vig r	Review code quality

Repository Commands

Command	Description
vigthoria repo push	Push project to Vigthoria Repo
vigthoria repo pull <name>	Pull project from Vigthoria Repo
vigthoria repo list	List your projects
vigthoria repo status	Check sync status
vigthoria repo share <name>	Generate shareable link
vigthoria repo delete <name>	Remove project from cloud
vigthoria repo clone <url>	Clone public project

Hub/Marketplace Commands

Command	Description
vigthoria hub discover	Interactive module discovery
vigthoria hub list	List all available modules
vigthoria hub search <query>	Semantic search for modules
vigthoria hub info <module>	Get module details

Command	Description
vigthoria hub activate <module>	Activate a module
vigthoria hub active	Show your active modules

Deploy Commands

Command	Description
vigthoria deploy preview	Deploy to preview URL
vigthoria deploy subdomain <name>	Deploy to name.vigthoria.io
vigthoria deploy custom <domain>	Deploy to custom domain
vigthoria deploy list	List all deployments
vigthoria deploy status [domain]	Check deployment status
vigthoria deploy verify <domain>	Verify DNS config
vigthoria deploy plans	Show hosting plans
vigthoria deploy remove <domain>	Remove deployment

Chat/Agent Options

Option	Description
-m, --model <model>	AI model to use
-p, --project <path>	Project context path
-a, --agent	Enable agent mode (chat only)
-r, --resume	Resume last session
-l, --local	Use local Ollama (no auth)
--auto-approve	Auto-approve agent actions

In-Session Commands

Command	Description
/help	Show all commands
/file <path>	Add file to context
/edit <path>	Start editing a file
/diff	Show pending changes
/apply	Apply pending changes
/model <name>	Switch AI model (or list available)
/agent	Toggle agent mode
/approve	Toggle auto-approve
/undo	Undo last file operation (agent mode)
/clear	Clear conversation
/compact	Compress old messages
/sessions	List saved sessions
/history	Show conversation history
/save	Save current session
/new	Start new session
/status	Show login & subscription status
/logout	Logout from Vigthoria
/exit	Exit (auto-saves)

Command	Description
/quit	Same as /exit

Auth Commands

Command	Description
vigthoria login	Authenticate with Vigthoria
vigthoria logout	Clear authentication
vigthoria status	Show account status

Config Commands

Command	Description
vigthoria config	Interactive configuration
vigthoria config --list	Show all settings
vigthoria config --set key=value	Set a value
vigthoria config --get key	Get a value
vigthoria config --reset	Reset to defaults
vigthoria init	Initialize project config

Global Options

Option	Description
-V, --version	Show version number
-h, --help	Show help

Workflows & Examples

Workflow 1: Feature Development with Agent Mode

1. Start agent mode

```
cd ~/projects/my-app
vigthoria agent
```

2. Let AI implement the feature

you > Create a user authentication system with JWT, including:

- Login and register endpoints
- Password hashing with bcrypt
- JWT token generation and validation
- Protected route middleware

3. Agent autonomously:

- # - Reads existing code
- # - Creates new files
- # - Installs dependencies
- # - Sets up the feature

Workflow 2: Quick Coding Session (Local)

```
# Start local mode (no internet needed)  
vigthoria chat --local -m code
```

you > Write a Python script to batch resize images

```
# Get instant help without cloud dependency
```

Workflow 3: Resume Yesterday's Work

```
# Resume where you left off  
vigthoria chat --resume
```

```
# Your previous conversation context is restored  
you > Continue implementing the caching layer we discussed
```

```
# AI remembers the context
```

Workflow 4: Code Review Before Commit

```
# Create a pre-commit hook  
#!/bin/bash  
# .git/hooks/pre-commit
```

```
for file in $(git diff --cached --name-only --diff-filter=ACM | grep '\.ts$'); do  
  # Review each file  
  result=$(vigthoria review "$file" -f json)  
  score=$(echo "$result" | jq '.score')  
  
  if [ "$score" -lt 70 ]; then  
    echo "Code review failed for $file (score: $score)"  
    exit 1  
  fi  
done
```

Workflow 5: Documentation Generation

```
# Generate documentation for a file  
vigthoria chat --model creative
```

you > /file src/core/engine.ts
✓ Added file to context

you > Generate comprehensive JSDoc documentation for all exported functions

```
# Apply the documentation  
you > /edit src/core/engine.ts
```

Workflow 6: Debugging Session

```
# Start debugging session  
vigthoria chat --model code
```

```
you > /file src/problematic.ts
you > /file src/tests/problematic.test.ts
```

you > The tests are failing with "undefined is not a function". Help me debug.

AI analyzes both files and provides debugging steps

Workflow 7: Project Scaffolding with Agent

```
vigthoria agent --auto-approve
```

you > Create a new REST API project with:

- TypeScript + Express
- PostgreSQL with Prisma ORM
- JWT authentication
- Docker configuration
- GitHub Actions CI/CD
- Jest testing setup

Agent creates entire project structure automatically

Workflow 8: Batch Processing

Fix style issues in all TypeScript files

```
find src -name "*.ts" -exec vigthoria fix {} -t style --apply \;
```

Generate review reports for a directory

```
for file in src/**/*.*ts; do
```

```
  vigthoria review "$file" -f markdown >> reviews/$(basename "$file" .ts).md
done
```

Troubleshooting

Common Issues

“Not authenticated” Error

Solution: Login again

```
vigthoria login
```

Or use local mode without authentication

```
vigthoria chat --local
```

“Subscription expired” Warning Your subscription needs renewal. Visit: <https://coder.vigthoria.io/settings> (Account section)

“API connection failed” Error

Check API status

```
vigthoria status
```

Try local mode as fallback

```
vigthoria chat --local
```

```
# Or try alternative endpoint
```

```
vigthoria config --set apiUrl=https://backup.coder.vigthoria.io
```

“Model not available” Error The requested model isn’t included in your subscription plan.

```
# Check available models
```

```
vigthoria status
```

```
# Use an available model
```

```
vigthoria chat --model mini
```

Ollama Connection Failed (Local Mode)

```
# Make sure Ollama is running
```

```
ollama serve
```

```
# Check if Ollama is accessible
```

```
curl http://localhost:11434/api/tags
```

```
# Pull a model if none available
```

```
ollama pull phi3:mini
```

Session Not Saving

```
# Sessions auto-save on /exit
```

```
# Force save manually:
```

```
/save
```

```
# Check session storage
```

```
ls ~/.vigthoria/sessions/
```

Agent Actions Not Working

```
# Check if agent mode is enabled
```

```
/agent
```

```
# Verify auto-approve status
```

```
/approve
```

Shell Completion Not Working

```
# Bash
```

```
source ~/.bash_completion.d/vigthoria
```

```
# Zsh - Add to ~/.zshrc
```

```
fpath=(~/.zsh/completion $fpath)
```

```
autoload -Uz compinit && compinit
```

Debug Mode

For detailed error information:

```
DEBUG=vigthoria* vigthoria chat
```

Reset Everything

```
# Clear config and auth
vigthoria logout
vigthoria config --reset
```

```
# Clear sessions
rm -rf ~/.vigthoria/sessions/
```

```
# Reinstall
npm uninstall -g vigthoria-cli
npm install -g vigthoria-cli
```

Report Issues

- GitHub: <https://github.com/vigthoria/vigthoria-cli/issues>
 - Discord: <https://discord.gg/vigthoria>
 - Email: support@vigthoria.io
-

FAQ

General

Q: Is Vigthoria CLI free? A: Basic features are free. Advanced models require a Pro or Ultra subscription. Local mode with Ollama is completely free.

Q: Does it work offline? A: Yes! Use `--local` mode with Ollama installed. Cloud features require internet.

Q: Is my code sent to the cloud? A: In cloud mode, code is sent to Vigthoria servers for AI processing. In local mode (`--local`), everything stays on your machine.

Q: How is this different from ChatGPT? A: Vigthoria CLI is specifically designed for coding:
- Project context awareness - Agent mode for autonomous coding - Integrated file editing - Code-optimized models - Terminal-native interface

Technical

Q: What models can I use? A: Depends on your subscription. Check with `vigthoria status`. In local mode, use Ollama models.

Q: Can I use it in CI/CD? A: Yes, use the `VIGTHORIA_TOKEN` environment variable for authentication.

Q: Does it support Windows? A: Yes, via Windows Terminal, PowerShell, or WSL.

Q: How do I update?

```
npm update -g @vigthoria/cli
```

Account

Q: Can I use my Vigthoria Coder account? A: Yes, same account works for CLI, Web IDE, and VS Code extension.

Q: How do I upgrade my subscription? A: Visit <https://coder.vigthoria.io/settings> to manage your subscription plan.

Getting Help

- **Documentation:** <https://docs.vigthoria.io/cli>
 - **Discord Community:** <https://discord.gg/vigthoria>
 - **GitHub Issues:** <https://github.com/vigthoria/vigthoria-cli/issues>
 - **Email Support:** support@vigthoria.io (Pro/Ultra subscribers)
-

Sophie — Autonomous Loop Engine

What is Sophie?

Sophie is Vigthoria's **Autonomous Loop Engine** — a persistent AI iteration system that takes a task, feeds it to AI models in a continuous loop, checks for completion at each step, and keeps iterating until the task is fully done or the maximum iteration limit is reached.

Think of Sophie as your tireless AI partner that doesn't stop at one response. Instead, she iterates: plan → execute → evaluate → refine → repeat — until the job is truly complete.

Installation

Sophie is installed as a system command when you set up the Vigthoria AGI platform:

```
# Sophie should already be available as a system command
```

```
sophie --help
```

```
# If not, create the symlink:
```

```
sudo ln -sf /var/www/agent-vigthoria/sophie_cli.py /usr/local/bin/sophie
```

```
sudo chmod +x /var/www/agent-vigthoria/sophie_cli.py
```

Quick Start

```
# Run a simple task
```

```
sophie run "Build a REST API with Express.js that has user authentication"
```

```
# Run with options
```

```
sophie run "Create a React dashboard with charts" \
```

```
--name "Dashboard Project" \
```

```
--max-iterations 25 \
```

```
--strategy incremental \
```

```
--workspace ./my-project
```

```
# Run from a task file
```

```
sophie run --file project-requirements.md --strategy test_driven
```

Sophie Commands

Command	Description
sophie run "task"	Create and immediately run an autonomous loop
sophie run --file <path>	Run task from a markdown/text file

Command	Description
<code>sophie status</code>	Show engine status (active loops, token usage)
<code>sophie list</code>	List all Sophie tasks
<code>sophie list --status running</code>	Filter tasks by status
<code>sophie detail <task_id></code>	Show detailed information about a task
<code>sophie history <task_id></code>	Show complete iteration history
<code>sophie pause <task_id></code>	Pause a running loop
<code>sophie resume <task_id></code>	Resume a paused loop
<code>sophie cancel <task_id></code>	Cancel and stop a loop
<code>sophie start <task_id></code>	Start a pending/draft task
<code>sophie squad <args></code>	Passthrough — delegates to Squad CLI

Run Options

`sophie run "task description" [OPTIONS]`

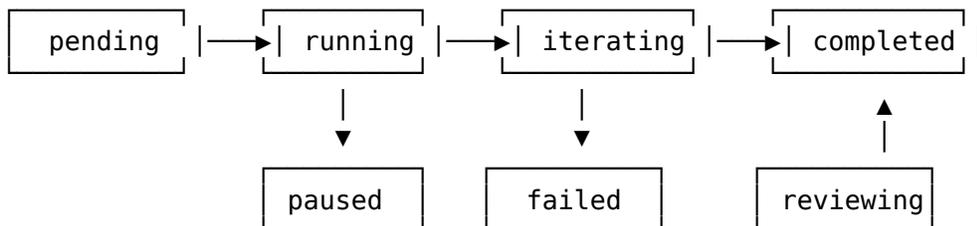
Options:

<code>--name NAME</code>	Human-friendly name for the task
<code>--max-iterations N</code>	Maximum iterations before auto-stop (default: 50)
<code>--strategy STRATEGY</code>	Iteration strategy to use
<code>--workspace PATH</code>	Working directory for file operations
<code>--file PATH</code>	Read task description from a file
<code>--priority LEVEL</code>	Task priority (low/normal/high/critical)

Iteration Strategies

Strategy	Description	Best For
<code>persistent</code>	Keep iterating with full context until completion signal	General-purpose tasks
<code>test_driven</code>	Write tests first, then iterate until all tests pass	Code with clear test criteria
<code>incremental</code>	Break task into phases, complete each sequentially	Large multi-step projects
<code>review_refine</code>	Generate, review, refine in cycles	Quality-critical deliverables

Task Lifecycle



Statuses: pending, running, iterating, paused, completed, failed, cancelled, reviewing

AI Backend (3-Tier Fallback)

Sophie uses a 3-tier AI backend for maximum reliability:

1. **Vigthoria Model Router** (localhost:4009) — intelligent model routing
2. **OpenRouter** — cloud fallback with 100+ models
3. **Ollama Local** — local fallback for offline/private work

Example: Building a Full Feature

```
# Create a complex task with test-driven strategy
sophie run "Build a complete user management system with:
- User registration with email verification
- Login with JWT tokens and refresh tokens
- Password reset via email
- Role-based access control (admin, user, moderator)
- User profile CRUD operations
- Rate limiting on auth endpoints
Write comprehensive tests for all endpoints." \
--name "User Management System" \
--strategy test_driven \
--max-iterations 30 \
--workspace ./backend

# Monitor progress
sophie status
sophie detail <task_id>

# Watch iteration history in real-time
watch -n 5 sophie history <task_id>
```

Sophie HTTP API

Sophie also exposes an HTTP API (requires admin authentication):

Endpoint	Method	Description
/api/sophie/status	GET	Engine status
/api/sophie/tasks	GET, POST	List or create tasks
/api/sophie/tasks/<id>	GET	Task details
/api/sophie/tasks/<id>/start	POST	Start a task
/api/sophie/tasks/<id>/pause	POST	Pause a task
/api/sophie/tasks/<id>/resume	POST	Resume a task
/api/sophie/tasks/<id>/cancel	POST	Cancel a task
/api/sophie/tasks/<id>/iterations	GET	Iteration history

Squad — Multi-Agent Team System

What is Squad?

Squad is Vigthoria's **Multi-Agent Team System** — an orchestration framework that spawns specialized AI agents (coordinator, architect, coder, tester, etc.) that collaborate on big projects. Each agent has its own task queue, terminal output, and communicates with other agents via a shared message bus.

Think of it like assembling a dream team of AI specialists: a coordinator plans the work, an architect designs the system, coders implement features, testers validate, and a reviewer ensures quality — all working in parallel.

Installation

```
# Squad should already be available as a system command  
squad --help
```

```
# If not, create the symlink:  
sudo ln -sf /var/www/agent-vigthoria/squad_cli.py /usr/local/bin/squad  
sudo chmod +x /var/www/agent-vigthoria/squad_cli.py
```

Quick Start

```
# Create a squad for a project  
squad create "Build an e-commerce platform with payment integration" \  
  --agents coordinator,architect,coder,tester,docs
```

```
# Deploy the squad (starts execution)  
squad deploy <squad_id>
```

```
# Monitor progress  
squad status <squad_id>  
squad agents <squad_id>
```

```
# Watch a specific agent's terminal  
squad terminal <squad_id> <agent_id>
```

Squad Commands

Command	Description
squad create "mission"	Create a new squad with a mission description
squad deploy <squad_id>	Deploy the squad — starts agent execution
squad status	Overview of all squads and the engine
squad status <squad_id>	Detailed status of a specific squad
squad list	List all squads
squad agents <squad_id>	Show agents in a squad with their roles and status
squad terminal <sid> <aid>	View an agent's terminal output (add --follow for live)
squad tasks <squad_id>	Show the task breakdown for a squad
squad msg <sid> <aid> "msg"	Send a direct message to a specific agent
squad broadcast <sid> "msg"	Broadcast a message to all agents in a squad
squad messages <squad_id>	View the squad's message log
squad pause <squad_id>	Pause squad execution
squad resume <squad_id>	Resume paused squad
squad disband <squad_id>	Cancel and disband the squad
squad roles	List all available agent roles

Create Options

squad create "mission description" [OPTIONS]

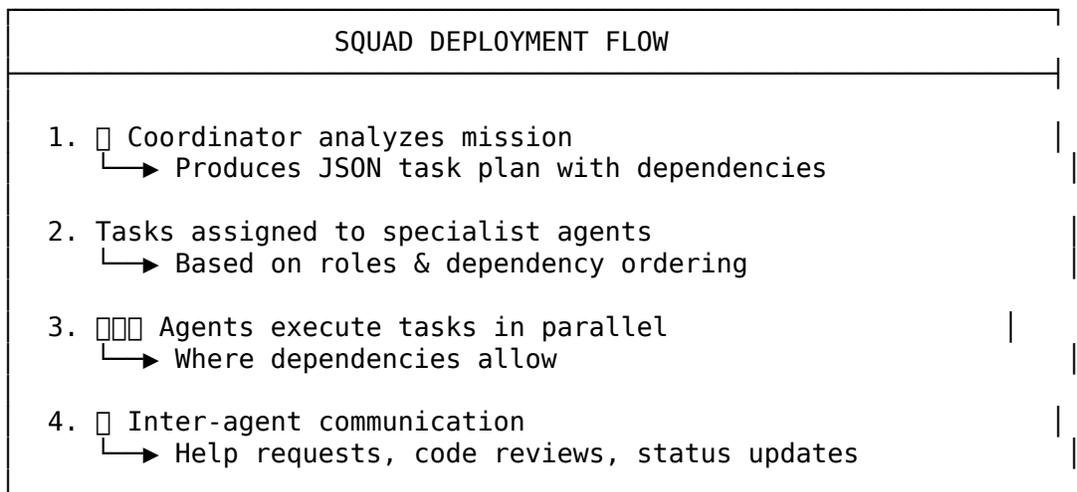
Options:

--agents ROLES	Comma-separated agent roles (default: coordinator,coder)
--name NAME	Human-friendly squad name
--workspace PATH	Working directory for file operations
--max-iterations N	Max iterations per agent (default: 20)
--priority LEVEL	Squad priority (low/normal/high/critical)

The 10 Agent Roles

Role	Emoji	Specialization
coordinator	👤	Breaks project into tasks, assigns work, monitors progress, integration checks
architect	🏗️	Designs system architecture, APIs, data models, technology decisions
coder	💻	Implements features, writes code, builds components
tester	🧪	Writes tests, validates features, creates test plans
debugger	🐛	Diagnoses and fixes bugs, analyzes error logs
docs	📄	Writes documentation, READMEs, API docs, user guides
reviewer	👁️	Code review, standards enforcement, quality gates
devops	🚀	Deployment, CI/CD pipelines, infrastructure setup
security	🛡️	Security audits, vulnerability scanning, auth design
data	🗄️	Database design, migrations, data pipelines, queries

Squad Deployment Flow



5. File Writer V2
 - ↳ Agents write actual files to project directory
6. Coordinator integration check
 - ↳ Verifies all pieces work together
7. Squad completes or fails with report

Example: Building a Complete Project

Create a full-stack squad

```
squad create "Build a task management app with:
- Next.js frontend with authentication
- Express API with PostgreSQL
- Real-time updates via WebSocket
- Docker deployment setup
- Full test coverage
- API documentation" \
--agents coordinator,architect,coder,tester,devops,docs \
--name "TaskManager Squad" \
--workspace ./task-manager
```

Deploy it

```
squad deploy <squad_id>
```

Watch the coordinator plan the work

```
squad terminal <squad_id> <coordinator_agent_id>
```

Check task breakdown

```
squad tasks <squad_id>
```

Watch a coder agent build features

```
squad terminal <squad_id> <coder_agent_id> --follow
```

Send guidance to the architect

```
squad msg <squad_id> <architect_agent_id> "Use microservices pattern, not monolith"
```

Broadcast a priority change to all agents

```
squad broadcast <squad_id> "Focus on the auth system first, it's blocking other features"
```

Monitor overall progress

```
watch -n 10 squad status <squad_id>
```

Inter-Agent Communication

Agents communicate via a shared message bus. Types of messages:

Message Type	Description
task_update	Agent reports task completion or progress
help_request	Agent asks another specialist for help
code_review	Agent sends code for review

Message Type	Description
dependency_complete	Signals that a blocking task is done
error_report	Agent reports an issue
broadcast	Human sends message to all agents

Squad HTTP API

Endpoint	Method	Description
/api/squad/status	GET	Engine status
/api/squad/squads	GET, POST	List or create squads
/api/squad/squads/<id>	GET	Squad details
/api/squad/squads/<id>/deploy	POST	Deploy a squad
/api/squad/squads/<id>/pause	POST	Pause a squad
/api/squad/squads/<id>/resume	POST	Resume a squad
/api/squad/squads/<id>/disband	POST	Disband a squad
/api/squad/squads/<id>/agents	GET	List agents
/api/squad/squads/<id>/messages	GET	Message log

Sophie + Squad Together

Sophie and Squad can work together. Use `sophie squad` to pass through to Squad commands:

From Sophie, delegate to Squad

```
sophie squad create "Build payment microservice" --agents coder,tester,security
```

Or use Sophie for single-task loops and Squad for multi-agent projects

```
sophie run "Optimize the database queries in /src/db/" --strategy review_refine
squad create "Build the entire frontend dashboard" --agents coordinator,coder,tester,docs
```

Quick Reference Card

VIGTHORIA CLI QUICK REFERENCE v2.0	
BASIC COMMANDS	
<code>vigthoria chat</code>	Interactive AI chat
<code>vigthoria agent</code>	Autonomous agent mode
<code>vigthoria chat --local</code>	Local mode (no auth needed)
<code>vigthoria chat --resume</code>	Resume last session
CODE COMMANDS	
<code>vigthoria edit <file></code>	Edit file with AI
<code>vigthoria generate "desc"</code>	Generate code from description
<code>vigthoria explain <file></code>	Explain code
<code>vigthoria fix <file></code>	Fix bugs/security/style
<code>vigthoria review <file></code>	Review code quality
SOPHIE – AUTONOMOUS LOOP ENGINE	
<code>sophie run "task"</code>	Start autonomous loop

sophie run --file task.md	Run from file		
sophie status	Engine status overview		
sophie list	List all tasks		
sophie detail <id>	Show task details		
sophie history <id>	Iteration history		
sophie pause/resume <id>	Control running loops		
sophie cancel <id>	Cancel a loop		
sophie squad <args>	Passthrough to Squad CLI		
SQUAD – MULTI-AGENT TEAM SYSTEM			
squad create "mission"	Create agent squad		
squad deploy <id>	Deploy squad to execute		
squad status [id]	Status overview or squad detail		
squad list	List all squads		
squad agents <id>	Show squad agents		
squad terminal <sid> <aid>	Agent terminal output		
squad msg <sid> <aid> "m"	Message an agent		
squad broadcast <sid> "m"	Broadcast to all agents		
squad tasks <id>	Show squad tasks		
squad pause/resume <id>	Control squad execution		
squad disband <id>	Cancel and disband squad		
squad roles	List available agent roles		
IN-SESSION COMMANDS			
/file <path>	Add file	/agent	Toggle agent
/edit <path>	Edit file	/approve	Toggle auto-approve
/model <name>	Switch model	/sessions	List sessions
/clear	Clear chat	/save	Save session
/compact	Compress	/history	Show history
/help	Show help	/exit	Exit (saves)
SHORTCUTS			
vig c = chat	vig a = agent	vig e = edit	vig g = gen
vig x = explain	vig f = fix	vig r = review	

Happy Coding with Vigthoria CLI! ☐

Built with ♥ by Vigthoria Technologies

Last Updated: January 2026