Vigthoria CLI Benutzerhandbuch

KI-gesteuerter Terminal-Codierassistent

Version 2.0.0

Inhaltsverzeichnis

Einführung

Was ist Vigthoria CLI?

Vigthoria CLI ist ein leistungsstarker Terminal-basierter Codierassistent, der die volle Leistungsfähigkeit der KI-Modelle von Vigthoria in Ihre Befehlszeile bringt. Egal ob Sie lokal, über SSH oder in einer Headless-Umgebung arbeiten - Vigthoria CLI bietet intelligente Code-Unterstützung, ohne das Terminal zu verlassen.

Hauptfunktionen

Funktion	Beschreibung
Interaktiver Chat	Natürlichsprachliche Codierunterstützung mit Kontextbewusstsein
Dateibearbeitung	KI-gesteuerte Codeänderungen mit Diff-Vorschau

Funktion	Beschreibung
Codegenerierung	Vollständigen Code aus Beschreibungen generieren
Code-Erklärung	Komplexen Code mit detaillierten Erklärungen verstehen
Fehlerbehebung	Automatische Problemerkennung und -behebung
Code-Review	Qualitätsanalyse mit umsetzbaren Vorschlägen
Multi-Modell-Unterstützung	Zugriff auf alle Vigthoria KI-Modelle je nach Abonnement
Projektkontext	Versteht Ihre Projektstruktur und Abhängigkeiten
♾ **Sophie Schleifen-Engine**	Autonome KI-Iterationsschleifen — Aufgabe geben, Sophie iteriert bis zur Fertigstellung
Squad Multi-Agent	Teams spezialisierter KI-Agenten für komplette Projektentwicklung

Für wen ist das?

- **Backend-Entwickler** - Die hauptsächlich in Terminal-Umgebungen arbeiten
- **DevOps-Ingenieure** - Für Skripterstellung und Automatisierungsaufgaben
- **Remote-Arbeiter** - SSH-Sitzungen zu entfernten Servern
- **Power-User** - Die tastaturgesteuerte Arbeitsabläufe bevorzugen
- **CI/CD-Integration** - Automatisierte Codegenerierung und Review

Voraussetzungen

Komponente	Minimum	Empfohlen
Node.js	18.0.0	20.x LTS
npm	8.x	10.x
OS	Linux, macOS, Windows	Linux/macOS
Internet	Erforderlich	Stabile Verbindung
Terminal	Beliebig	iTerm2, Warp, Kitty

Installation

Schnellinstallation (Empfohlen)

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

Dieses Skript wird: 1. Die Node.js-Version prüfen 2. Vigthoria CLI global installieren 3. Shell-Vervollständigungen einrichten 4. Befehlsverknüpfungen erstellen

npm Globale Installation

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

Manuelle Installation

```
# Von Vigthoria Market klonen
git clone https://market.vigthoria.io/vigthoria/vigthoria-cli.git
cd vigthoria-cli
```

```
# Oder von Vigthoria Community
git clone https://community.vigthoria.io/repos/vigthoria-cli.git
cd vigthoria-cli

# Abhängigkeiten installieren
npm install

# Kompilieren
npm run build

# Global verknüpfen
npm link
```

Installation überprüfen

```
vigthoria --version
# Ausgabe: 1.0.2

vigthoria --help
# Zeigt alle verfügbaren Befehle
```

Schnellstartanleitung

Schritt 1: Anmelden

```
vigthoria login
```

Wählen Sie Ihre Anmeldemethode: - **E-Mail & Passwort** - Geben Sie Ihre Vigthoria-Kontodaten ein - **API-Token** - Verwenden Sie ein Token aus Ihrem Vigthoria Coder Dashboard - **Browser-Anmeldung** - OAuth-Flow (öffnet Browser)

Schritt 2: Status prüfen

```
vigthoria status
```

Dies zeigt: - Kontoinformationen - Abonnementplan - Verfügbare Modelle - API-Konnektivität

Schritt 3: Mit dem Codieren beginnen

```
# Interaktiven Chat starten
vigthoria chat

# Oder den kurzen Alias verwenden
vig c
```

Schritt 4: Ihr erstes KI-Gespräch

```
you › Erstelle eine Funktion zur E-Mail-Validierung in TypeScript

vigthoria ›
Hier ist eine robuste E-Mail-Validierungsfunktion:
```

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

// Erweiterte Version mit detaillierter Validierung
interface EmailValidationResult {
  valid: boolean;
  reason?: string;
}

function validateEmailDetailed(email: string): EmailValidationResult {
  if (!email) {
    return { valid: false, reason: 'E-Mail ist erforderlich' };
  }

  if (email.length > 254) {
    return { valid: false, reason: 'E-Mail zu lang' };
  }

   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: 'Ungültiges E-Mail-Format' };
  }

  return { valid: true };
}
```

Authentifizierung

Anmeldemethoden

E-Mail & Passwort

```bash
vigthoria login
# Wählen Sie: E-Mail & Passwort
# Geben Sie Ihre E-Mail und Ihr Passwort ein
```

API-Token Holen Sie Ihr Token von: `https://coder.vigthoria.io/settings` (API-Schlüssel Bereich)

```bash
# Interaktiv
vigthoria login
# Wählen Sie: API Token
# Fügen Sie Ihr Token ein

# Oder direkt
vigthoria login --token IHR_API_TOKEN
```

Umgebungsvariable Für CI/CD oder automatisierte Umgebungen:

```
export VIGTHORIA_TOKEN="ihr-api-token"
vigthoria status  # Verwendet das Umgebungstoken
```

Authentifizierungsstatus prüfen

```
vigthoria status
```

Ausgabe:

```
═══ Kontostatus ═══

Konto:
  E-Mail: user@example.com
  Benutzer-ID: usr_abc123

Abonnement:
  Plan: PRO
  Status: Aktiv
  Läuft ab: 31.12.2026 (365 Tage)

Verfügbare Modelle:
  • vigthoria-fast - Schnelle Antworten (1.1B)
  • vigthoria-mini - Ausgewogen (3.8B)
  • vigthoria-code - Code-Spezialist (8B)
  • vigthoria-creative - Kreativ (9B)

API-Status:
  Server: Online
  Endpunkt: https://coder.vigthoria.io
```

Abmelden

```
vigthoria logout
```

Interaktiver Chat-Modus

Der Chat-Modus ist die primäre Methode zur Interaktion mit der Vigthoria KI. Er behält den Gesprächskontext bei und versteht Ihr Projekt.

Chat starten

```
vigthoria chat
# oder
vig c

# Mit spezifischem Modell
vigthoria chat --model pro

# Mit Projektkontext
vigthoria chat --project /pfad/zum/projekt
```

Chat-Befehle

Verwenden Sie im Chat-Modus diese speziellen Befehle:

Befehl	Beschreibung
`/help`	Alle verfügbaren Befehle anzeigen
`/file <pfad>`	Dateiinhalt zum Gespräch hinzufügen
`/edit <pfad>`	Bearbeitung einer bestimmten Datei starten
`/diff`	Ausstehende Änderungen anzeigen
`/apply`	Ausstehende Änderungen auf Dateien anwenden
`/model <name>`	KI-Modell wechseln (oder verfügbare auflisten)
`/agent`	Agent-Modus umschalten
`/approve`	Auto-Genehmigung umschalten
`/undo`	Letzte Dateioperation rückgängig machen (Agent-Modus)
`/clear`	Gesprächsverlauf löschen
`/compact`	Ältere Nachrichten komprimieren
`/sessions`	Gespeicherte Sitzungen auflisten
`/history`	Gesprächsverlauf anzeigen
`/save`	Aktuelle Sitzung speichern
`/new`	Neue Sitzung starten
`/status`	Login- & Abonnement-Status anzeigen
`/logout`	Von Vigthoria abmelden
`/exit`	Beenden (speichert automatisch)
`/quit`	Gleich wie /exit

Beispiel-Chat-Sitzung

```
╔═══════════════════════════════════════════════════════╗
  VIGTHORIA CLI - KI-gesteuerter Codierassistent
╚═══════════════════════════════════════════════════════╝

┌─────────────────────────────────────────────────────┐
Vigthoria Chat
Modell: vigthoria-code
Plan: PRO
Projekt: /home/user/mein-projekt
└─────────────────────────────────────────────────────┘

Geben Sie Ihre Nachricht ein oder /help für Befehle. Strg+C zum Beenden.

you › /file src/api.ts
✓ src/api.ts zum Kontext hinzugefügt (245 Zeilen)

you › Füge Rate-Limiting zum Authentifizierungsendpunkt hinzu

vigthoria ›
Ich werde Rate-Limiting zu Ihrem Authentifizierungsendpunkt hinzufügen...

[... detaillierte Antwort mit Code ...]

you › /edit src/api.ts
Welche Änderungen möchten Sie vornehmen?
> Füge den Rate-Limiting-Code hinzu, den du gerade gezeigt hast
```

```
[Diff-Vorschau]
Änderungen anwenden? (J/n) j
✓ Änderungen auf src/api.ts angewendet
```

Agent-Modus

Der Agent-Modus ist das leistungsstärkste Feature der Vigthoria CLI Er ermöglicht der KI, autonom zu:

- **Dateien lesen** - Beliebige Dateien in Ihrem Projekt untersuchen
- ✏ **Dateien schreiben** - Dateien erstellen oder modifizieren
- **Suchen** - Grep-Muster und Glob-Dateien
- **Ausführen** - Shell-Befehle ausführen
- **Durchsuchen** - Verzeichnisse auflisten
- **Git** - Git-Befehle ausführen

Agent-Modus starten

```
# Direkter Agent-Befehl
vigthoria agent

# Oder mit Shortcut
vig a

# Oder im Chat aktivieren
vigthoria chat --agent
vigthoria chat -a
```

Agent-Optionen

Option	Beschreibung	Standard
`-m, --model <model>`	Zu verwendendes KI-Modell	`balanced`
`-p, --project <path>`	Projektkontext-Pfad	Aktuelles Verzeichnis
`-l, --local`	Lokales Ollama verwenden	`false`
`--auto-approve`	Alle Aktionen automatisch genehmigen	`false`
`-r, --resume`	Letzte Sitzung fortsetzen	`false`

Verfügbare Tools

Die KI kann diese Tools im Agent-Modus verwenden:

Tool	Beschreibung	Beispiel-Verwendung
`read_file`	Dateiinhalt mit Zeilenbereich lesen	“Lies Zeilen 1-50 von index.ts”
`write_file`	Datei erstellen oder überschreiben	“Erstelle eine neue config.ts Datei”
`edit_file`	Suchen & ersetzen in Datei	“Ändere den Funktionsnamen”
`bash`	Shell-Befehle ausführen	“Führe npm install express aus”

Tool	Beschreibung	Beispiel-Verwendung
`grep`	Muster in Dateien suchen	“Finde alle TODO-Kommentare”
`list_dir`	Verzeichnisinhalt auflisten	“Zeige Dateien in src/”
`glob`	Dateien nach Muster finden	”Finde alle *.test.ts Dateien”
`git`	Git-Befehle ausführen	“git status”

Berechtigungssystem

Standardmäßig **fragt der Agent-Modus nach Berechtigung**, bevor: - Dateien geschrieben oder bearbeitet werden - Shell-Befehle ausgeführt werden - Zerstörerische Änderungen vorgenommen werden

```
⚠ Die KI möchte write_file verwenden
   Datei: src/utils/helper.ts
   Inhalt: [Vorschau angezeigt]

Aktion erlauben? [j/N] j
✓ write_file: Erfolgreich
```

Auto-Genehmigungs-Modus

⚠ **WARNUNG: GEFÄHRLICH** - Nur in vertrauenswürdigen Umgebungen verwenden!

Der Auto-Genehmigungs-Modus erlaubt alle KI-Aktionen ohne Bestätigung:

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

Oder innerhalb einer Sitzung umschalten:

```
you › /approve
✓ Auto-approve: AN - Alle Aktionen werden automatisch ausgeführt
```

Beispiel Agent-Sitzung

```
╔═══════════════════════════════════════════════════════╗
║   VIGTHORIA CLI - KI-gestützter Codierassistent       ║
║   Version 1.4.1                                       ║
╚═══════════════════════════════════════════════════════╝

[Agent-Modus] Modell: vigthoria-code | Auto-approve: AUS

you › Erstelle einen Express-Server mit TypeScript und füge Benutzerauthentifizierung hinzu

vigthoria › Ich erstelle einen Express-Server mit Authentifizierung für dich. Lass mich:
1. Deine aktuelle Projektstruktur prüfen
2. Die notwendigen Dateien erstellen
3. Abhängigkeiten einrichten

⚙ Ausführung: list_dir
   Pfad: .
✓ list_dir: package.json, src/, tsconfig.json

⚙ Ausführung: read_file
   Datei: package.json, Zeilen: 1-30
✓ read_file: Erfolgreich (25 Zeilen)
```

```
⚙ Ausführung: write_file
   Datei: src/server.ts
   Express-Server mit Authentifizierung wird erstellt...
Erlauben? [j/N] j
✓ write_file: Erfolgreich

⚙ Ausführung: write_file
   Datei: src/middleware/auth.ts
Erlauben? [j/N] j
✓ write_file: Erfolgreich

⚙ Ausführung: bash
  Befehl: npm install express jsonwebtoken bcryptjs @types/express @types/jsonwebtoken
Erlauben? [j/N] j
✓ bash: Erfolgreich

Ich habe deinen Express-Server mit JWT-Authentifizierung erstellt!

Erstellte Dateien:
- src/server.ts - Hauptserver-Datei
- src/middleware/auth.ts - Authentifizierungs-Middleware
- src/routes/auth.ts - Login/Register-Routen

Führe `npm run dev` aus, um den Server zu starten.
```

Agent- vs Chat-Modus Vergleich

Feature	Chat-Modus	Agent-Modus
Konversation	✓	✓
Dateikontext	Manuell (`/file`)	Automatisch
Dateien lesen	✓	✓
Dateien schreiben	Via `/edit`	Direkt
Befehle ausführen	✗	✓
Autonome Aktionen	✗	✓
Am besten für	Q&A, Diskussion	Implementierung

Sitzungsverwaltung

Sitzungen ermöglichen es Ihnen, **Konversationen zu speichern und fortzusetzen**, wobei der Kontext über mehrere Arbeitssitzungen hinweg erhalten bleibt.

Wie Sitzungen funktionieren

- Jede Chat/Agent-Sitzung hat eine eindeutige ID
- Sitzungen werden in `~/.vigthoria/sessions/` gespeichert
- Sitzungen sind projektspezifisch (basierend auf dem Arbeitsverzeichnis)
- Automatisch gespeichert beim Beenden mit `/exit`

Sitzungen anzeigen

```
you › /sessions

═══ Gespeicherte Sitzungen ═══
m1abc123-xyz [agent] (aktuell)
  /home/user/my-project - 13.01.2026, 20:30:00
m1def456-uvw
  /home/user/other-project - 12.01.2026, 15:15:00
m1ghi789-rst [agent]
  /home/user/my-project - 11.01.2026, 14:00:00
```

Sitzung fortsetzen

```
# Letzte Sitzung für aktuelles Projekt fortsetzen
vigthoria chat --resume
vigthoria chat -r

# Oder im Agent-Modus
vigthoria agent --resume
```

Sitzungen speichern

Sitzungen werden automatisch beim `/exit` gespeichert. Manuell erzwingen:

```
you › /save
✓ Sitzung gespeichert: m1abc123-xyz
```

Neue Sitzung starten

```
you › /new
✓ Vorherige Sitzung gespeichert
✓ Neue Sitzung: m1jkl012-mno
```

Konversationsverlauf anzeigen

```
you › /history

═══ Konversationsverlauf ═══
1. you: Wie erstelle ich eine REST API...
2. vigthoria: So erstellst du eine REST API...
3. you: Kannst du Authentifizierung hinzufügen...
4. vigthoria: Ich füge JWT-Authentifizierung hinzu...
```

Lange Sitzungen komprimieren

Wenn Konversationen lang werden, alte Nachrichten komprimieren:

```
you › /compact
✓ 12 Nachrichten zu Zusammenfassung komprimiert
```

Dies fasst ältere Nachrichten zusammen, um die Kontextgröße zu reduzieren, wobei wichtige Informationen erhalten bleiben.

Sitzungsdatei-Speicherort

```
~/.vigthoria/sessions/
├── m1abc123-xyz.json
├── m1def456-uvw.json
└── m1ghi789-rst.json
```

`vigthoria edit <datei> [optionen]`

Optionen: - `-i, --instruction <text>` - Bearbeitungsanweisung - `-m, --model <model>` - Zu verwendendes KI-Modell

Beispiele:

```
# Interaktive Bearbeitung
vigthoria edit src/utils.ts

# Mit Anweisung
vigthoria edit src/api.ts -i "Füge Fehlerbehandlung zu allen async-Funktionen hinzu"

# Mit spezifischem Modell
vigthoria edit src/complex.ts -m pro
```

Code-Probleme beheben

`vigthoria fix <datei> [optionen]`

Optionen: - `-t, --type <typ>` - Behebungstyp: `bugs`, `style`, `security`, `performance` - `--apply` - Behebungen automatisch ohne Bestätigung anwenden

Beispiele:

```
# Bugs finden und beheben
vigthoria fix src/index.ts -t bugs

# Sicherheitsprobleme mit Auto-Apply beheben
vigthoria fix src/auth.ts -t security --apply

# Performance-Probleme beheben
vigthoria fix src/heavy-computation.ts -t performance
```

Code erklären

`vigthoria explain <datei> [optionen]`

Optionen: - `-l, --lines <bereich>` - Zeilenbereich (z.B. `1-50`) - `-d, --detail <stufe>` - `brief`, `normal`, `detailed`

Beispiele:

```
# Gesamte Datei erklären
vigthoria explain src/algorithm.ts

# Bestimmte Zeilen erklären
vigthoria explain src/utils.ts -l 50-100

# Detaillierte Erklärung
vigthoria explain src/complex.ts -d detailed
```

Dateioperationen

Datei bearbeiten

`vigthoria edit <datei> [optionen]`

Optionen: - `-i, --instruction <text>` - Bearbeitungsanweisung - `-m, --model <model>` - Zu verwendendes KI-Modell

Beispiele:

```
# Interaktive Bearbeitung
vigthoria edit src/utils.ts

# Mit Anweisung
vigthoria edit src/api.ts -i "Füge Fehlerbehandlung zu allen async-Funktionen hinzu"

# Mit spezifischem Modell
vigthoria edit src/complex.ts -m pro
```

Bearbeitungs-Workflow

1. Dateiinhalt wird geladen
2. Sie beschreiben die Änderungen
3. KI generiert den modifizierten Code
4. Diff wird zur Überprüfung angezeigt
5. Sie genehmigen oder lehnen Änderungen ab
6. Backup wird automatisch erstellt

Code-Probleme beheben

`vigthoria fix <datei> [optionen]`

Optionen: - `-t, --type <typ>` - Behebungstyp: `bugs`, `style`, `security`, `performance` - `--apply` - Korrekturen automatisch anwenden ohne Bestätigung

Beispiele:

```
# Bugs finden und beheben
vigthoria fix src/index.ts -t bugs

# Sicherheitsprobleme mit Auto-Apply beheben
vigthoria fix src/auth.ts -t security --apply

# Performance-Probleme beheben
vigthoria fix src/heavy-computation.ts -t performance
```

Code erklären

`vigthoria explain <datei> [optionen]`

Optionen: - `-l, --lines <bereich>` - Zeilenbereich (z.B. `1-50`) - `-d, --detail <level>` - `brief`, `normal`, `detailed`

Beispiele:

```
# Gesamte Datei erklären
vigthoria explain src/algorithm.ts

# Spezifische Zeilen erklären
vigthoria explain src/utils.ts -l 50-100

# Detaillierte Erklärung
vigthoria explain src/complex.ts -d detailed
```

Codegenerierung

Code aus Beschreibung generieren

`vigthoria generate <beschreibung> [optionen]`

Optionen: - `-l, --language <sprache>` - Zielsprache (Standard: typescript) - `-o, --output <datei>` - Ausgabedateipfad - `-m, --model <model>` - Zu verwendendes KI-Modell

Unterstützte Sprachen

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

Beispiele

```
# TypeScript-Code generieren
vigthoria generate "REST API Endpunkt für Benutzerauthentifizierung"

# Python mit Ausgabedatei generieren
vigthoria generate "Datenbank-Migrationsskript für PostgreSQL" -l python -o migrate.py

# Mit spezifischem Modell generieren
vigthoria generate "Hochleistungs-Caching-Schicht" -l rust -m pro
```

Code-Review & Fehlerbehebung

Code-Review

`vigthoria review <datei> [optionen]`

Optionen: - `-f, --format <format>` - `text`, `json`, `markdown`

Review-Ausgabe

```
═══ Reviewing: src/api.ts ═══
Sprache: typescript | Zeilen: 245

Qualitätsbewertung: 78/100
████████████████████░░░░░░░░

═══ Probleme (4) ═══

✗ [security] Zeile 45: Potenzielle SQL-Injection-Schwachstelle
⚠ [performance] Zeile 112: Unnötiges Re-Render in Schleife
⚠ [style] Zeile 67: Magische Zahl sollte eine benannte Konstante sein
 [info] Zeile 189: Optionale Verkettung in Betracht ziehen

═══ Vorschläge ═══

1. Eingabevalidierung zur createUser-Funktion hinzufügen
2. Rate-Limiting für öffentliche Endpunkte implementieren
3. Umgebungsvariablen für Konfigurationswerte verwenden
4. JSDoc-Kommentare zu exportierten Funktionen hinzufügen
```

Konfiguration

Interaktive Konfiguration

```
vigthoria config
```

Einzelwerte setzen

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

Werte abrufen

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

Alle Einstellungen auflisten

```
vigthoria config --list
```

Auf Standardwerte zurücksetzen

```
vigthoria config --reset
```

Projekt initialisieren

Projektspezifische Konfiguration erstellen:

```
cd /pfad/zum/projekt
vigthoria init
```

Verfügbare KI-Modelle

Modell	Größe	Tokens	Optimal für	Pläne
`vigthoria-fast`	1.1B	4K	Schnelle Antworten, einfache Aufgaben	Free, Pro, Ultra
`vigthoria-mini`	3.8B	8K	Ausgewogene Leistung	Free, Pro, Ultra
`vigthoria-code`	8B	16K	Codegenerierung, Debugging	Pro, Ultra
`vigthoria-creative`	9B	16K	Kreatives Schreiben, Dokumentation	Pro, Ultra
`vigthoria-pro`	32B	32K	Komplexe Aufgaben, Architektur	Ultra
`vigthoria-ultra`	70B	64K	Maximale Fähigkeiten	Ultra

Modellauswahl

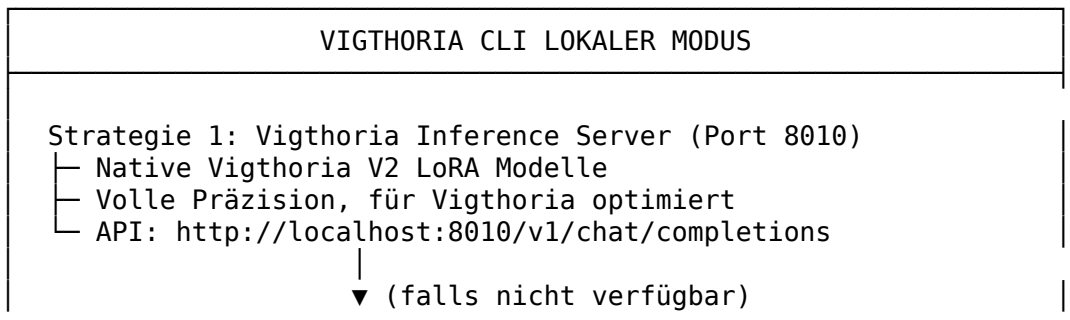
```
# Im Chat
/model code

# Befehlszeile
vigthoria chat --model pro
vigthoria generate "..." --model ultra
```

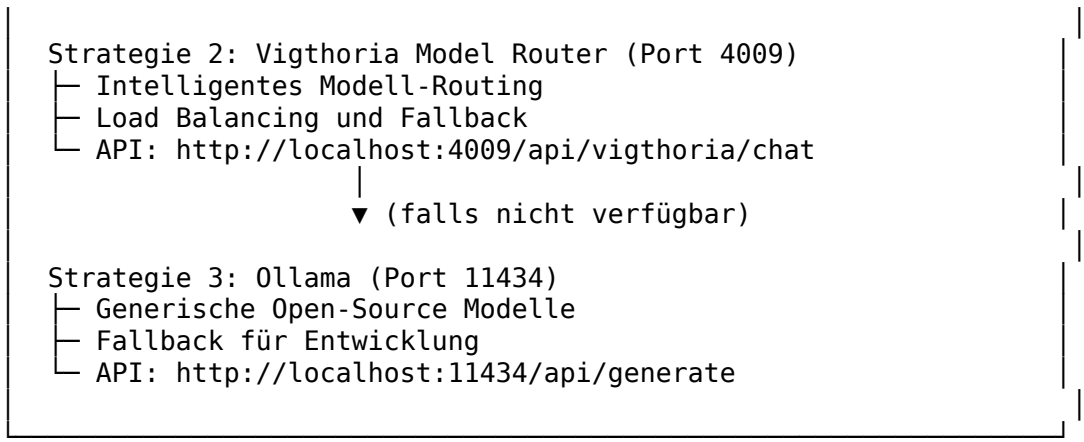
Lokaler Modus mit Vigthoria Inference

Der lokale Modus ermöglicht die Nutzung von Vigthoria CLI mit den **nativen Vigthoria LLMs** über den Vigthoria Inference Server. Dies gibt Ihnen Zugriff auf dieselben leistungsstarken Modelle wie in der Cloud, die lokal auf Ihrer Maschine laufen.

API-Verbindungsstrategie

Bei Verwendung von `--local` versucht Vigthoria CLI, sich in folgender Reihenfolge zu verbinden:





Voraussetzungen

Installieren Sie Vigthoria Inference von Vigthoria Market:

```
# Von Vigthoria Market klonen
git clone https://market.vigthoria.io/vigthoria/vigthoria-inference.git
cd vigthoria-inference

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

# Abhängigkeiten installieren
pip install -r requirements.txt

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

OpenAI-kompatible API

Vigthoria Inference Server bietet eine **OpenAI-kompatible API**:

```
# Direkter API-Aufruf an Vigthoria Inference
curl http://localhost:8010/v1/chat/completions \
  -H "Content-Type: application/json" \
  -d '{
    "model": "vigthoria-v2-code-8b",
    "messages": [{"role": "user", "content": "Schreibe Hello World in Python"}],
    "max_tokens": 512,
    "temperature": 0.7
  }'
```

Verfügbare Native Vigthoria Modelle

Vigthoria Inference bietet Zugriff auf alle nativen Vigthoria LLMs:

Modell	Größe	Beschreibung
vigthoria-mini-0.6b	0.6B	Ultra-schnelle Antworten
vigthoria-fast-1.7b	1.7B	Schnelles Streaming

Modell	Größe	Beschreibung
vigthoria-balanced-4b	4B	Allzweck
vigthoria-code-v2-8b	8B	Code-Spezialist
vigthoria-creative-9b-v4	9B	Kreatives Schreiben
vigthoria-music-master-4b	4B	Musikproduktion

Lokalen Modus verwenden

```
# Chat mit Vigthoria Inference starten
vigthoria chat --local
vigthoria chat -l

# Agent-Modus mit lokaler Inferenz
vigthoria agent --local

# Mit spezifischem Vigthoria-Modell
vigthoria chat --local --model code
vigthoria chat -l -m balanced
```

Lokale Modellzuordnung

Bei Verwendung von `--local` verbindet sich Vigthoria CLI mit Vigthoria Inference (Port 8010):

Alias	Vigthoria Modell	Größe
`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

Vorteile des Lokalen Modus

Vorteil	Beschreibung
Datenschutz	Code verlässt nie Ihre Maschine
Native Modelle	Volle Leistung der Vigthoria LLMs
Offline	Funktioniert ohne Internet
Keine Auth	Keine Anmeldung erforderlich
Geschwindigkeit	Geringe Latenz für lokale Inferenz
Kostenlos	Keine API-Kosten

Systemanforderungen

Modell	VRAM benötigt	Empfohlene GPU
Mini (0.6B)	1 GB	Jede GPU
Fast (1.7B)	2 GB	GTX 1060+
Balanced (4B)	4 GB	RTX 3060+
Code (8B)	8 GB	RTX 3070+

Modell	VRAM benötigt	Empfohlene GPU
Creative (9B)	10 GB	RTX 4080+

Hinweis: Mit 4-Bit-Quantisierung werden die VRAM-Anforderungen um ~40% reduziert.

Beispiel für Lokale Sitzung

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

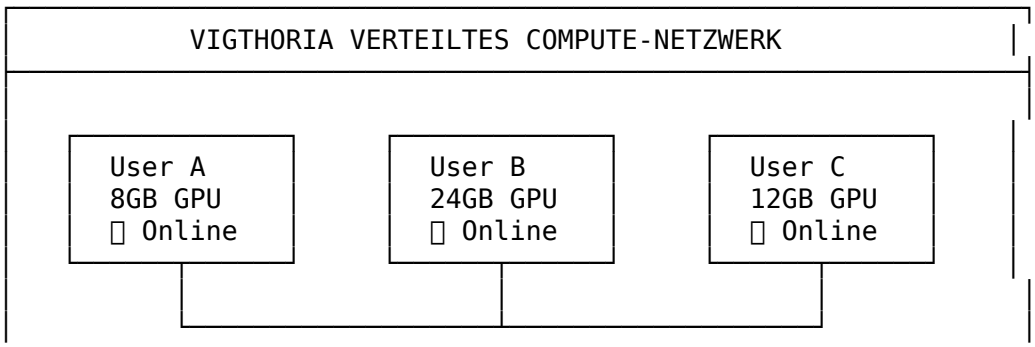
```
╔═══════════════════════════════════════════════════════╗
  VIGTHORIA CLI - KI-gesteuerter Codierassistent
  Version 1.4.1
╚═══════════════════════════════════════════════════════╝

[Lokaler Modus] Modell: vigthoria-code-v2-8b | Inferenz: localhost:8010

you › Schreibe eine Python-Funktion zum Umkehren eines Strings

vigthoria › Hier ist eine Python-Funktion zum Umkehren eines Strings:

def reverse_string(s: str) -> str:
    """Kehrt den Eingabestring um."""
    return s[::-1]

# Alternative mit reversed()
def reverse_string_alt(s: str) -> str:
    return ''.join(reversed(s))
```

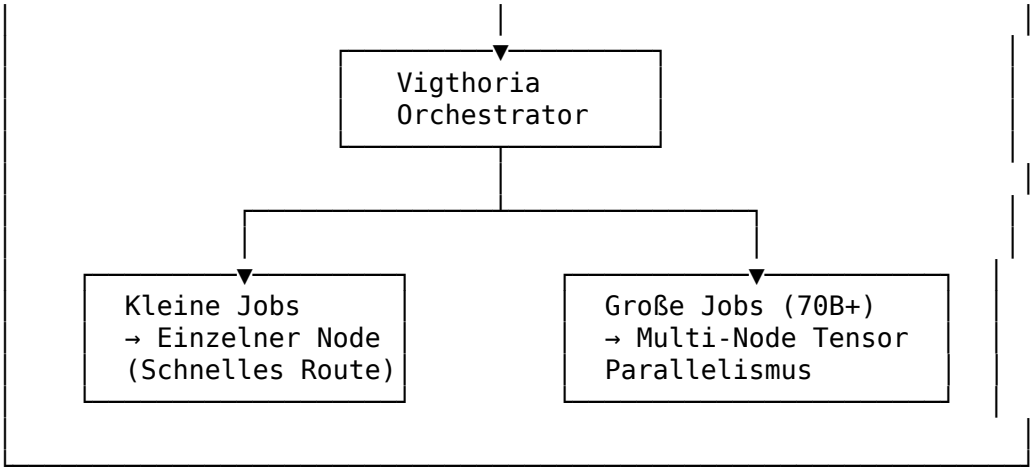
Verteiltes Computing (Demnächst)

Vision: Benutzer-gestützte KI

Vigthoria baut ein verteiltes Compute-Netzwerk auf, das Benutzern ermöglicht: 1. **GPU-Leistung beisteuern** - Teilen Sie Ihre lokale GPU mit dem Vigthoria-Netzwerk 2. **Belohnungen verdienen** - Erhalten Sie Vigthoria Credits für Compute-Beiträge 3. **Größere Modelle nutzen** - Führen Sie 32B, 70B und 128B Modelle über mehrere GPUs aus

So funktioniert es





Befehle (Demnächst)

```
# Compute-Sharing aktivieren
vigthoria compute enable

# Beitragsstatistiken prüfen
vigthoria compute status

# Compute-Limits konfigurieren
vigthoria compute config --max-vram 80%

# Verdiente Credits anzeigen
vigthoria credits
```

Belohnungssystem

Aktion	Verdiente Credits
1 Stunde Compute-Beitrag	10 Credits
Kleinen Inference-Job verarbeiten	1 Credit
Großen Inference-Job verarbeiten	5 Credits
An verteiltem 70B teilnehmen	20 Credits

Credits verwenden für

- Kostenloser Zugang zu Premium-Modellen
- Erweiterte Kontextfenster (32K, 64K, 128K)
- Prioritätswarteschlange für Inferenz
- Speicherplatz auf Vigthoria Market

Vigthoria Repository

Ihr persönliches Cloud-Repository zum Speichern und Teilen von Projekten. Laden Sie Ihren Code in Vigthorias sichere Cloud-Speicherung hoch und greifen Sie von überall darauf zu.

Projekt hochladen (Push)

```
# Aktuelles Verzeichnis hochladen
vigthoria repo push

# Bestimmten Pfad hochladen
vigthoria repo push /pfad/zum/projekt

# Mit Sichtbarkeitseinstellung hochladen
vigthoria repo push --visibility private    # Nur Sie können es sehen
vigthoria repo push --visibility public     # Jeder kann es sehen
vigthoria repo push --visibility restricted # Nur mit Link

# Überschreiben erzwingen
vigthoria repo push --force
```

Projekt herunterladen (Pull)

```
# Projekt aus Ihrem Repo herunterladen
vigthoria repo pull mein-projekt

# In bestimmtes Verzeichnis herunterladen
vigthoria repo pull mein-projekt --output /pfad/zum/ziel

# Bestehendes Verzeichnis überschreiben
vigthoria repo pull mein-projekt --force
```

Projekte auflisten

```
# Alle Projekte auflisten
vigthoria repo list
# oder
vigthoria repo ls

# Nach Sichtbarkeit filtern
vigthoria repo list --visibility private
```

Sync-Status prüfen

```
# Sync-Status des aktuellen Projekts prüfen
vigthoria repo status
```

Projekte teilen

```
# Teilbaren Link generieren (Standard 7 Tage)
vigthoria repo share mein-projekt

# Benutzerdefinierte Ablaufzeit
vigthoria repo share mein-projekt --expires 24h
vigthoria repo share mein-projekt --expires 30d
```

Projekt löschen

```
# Aus Cloud entfernen (lokale Dateien unverändert)
vigthoria repo delete mein-projekt
# oder
vigthoria repo rm mein-projekt
```

Öffentliche Projekte klonen

```
# Von Vorschau-URL klonen
vigthoria repo clone https://coder.vigthoria.io/preview/123

# Nach Projekt-ID klonen
vigthoria repo clone 123
```

Vigthoria Hub (API-Marktplatz)

Entdecken und aktivieren Sie leistungsstarke KI-APIs aus Vigthorias Modul-Marktplatz. Pay-as-you-go-Abrechnung für Musikgenerierung, Video-KI, Stimmenklonen und mehr.

Module entdecken

```
# Interaktiver Entdeckungs-Assistent
vigthoria hub discover
# oder
vig hub d
```

Module suchen

```
# Semantische Suche nach Modulen
vigthoria hub search "Hintergrundmusik für meine App generieren"
vigthoria hub search "KI-Spracherzählung erstellen"
```

Alle Module auflisten

```
# Alle verfügbaren Module anzeigen
vigthoria hub list
# oder
vigthoria hub ls

# Nach Kategorie filtern
vigthoria hub list --category music
vigthoria hub list --category video
vigthoria hub list --category voice
```

Modul-Details anzeigen

```
# Detaillierte Informationen zu einem Modul
vigthoria hub info vigthoria-music-ai
vigthoria hub details vigthoria-voice-clone
```

Modul aktivieren

```
# Für Ihren API-Key aktivieren (aktiviert Pay-as-you-go)
vigthoria hub activate vigthoria-music-ai
vigthoria hub enable vigthoria-voice-clone
```

Aktive Module anzeigen

```
# Ihre aktuell aktiven Module anzeigen
vigthoria hub active
```

Bereitstellung & Hosting

Stellen Sie Ihre Projekte mit einem Befehl auf Vigthorias Hosting-Infrastruktur bereit.

Vorschau-Bereitstellung

```
# Auf kostenlose Vorschau-URL bereitstellen
vigthoria deploy preview
```

Sie erhalten eine URL wie: `https://preview-abc123.vigthoria.io`

Subdomain-Bereitstellung

```
# Auf ihrname.vigthoria.io bereitstellen
vigthoria deploy subdomain meinprojekt

# Ergebnis: https://meinprojekt.vigthoria.io
```

Eigene Domain

```
# Auf Ihre eigene Domain bereitstellen
vigthoria deploy custom meineapp.com

# DNS-Konfiguration überprüfen
vigthoria deploy verify meineapp.com
```

Bereitstellungen verwalten

```
# Alle Bereitstellungen auflisten
vigthoria deploy list

# Bereitstellungsstatus prüfen
vigthoria deploy status meinprojekt.vigthoria.io

# Bereitstellung entfernen
vigthoria deploy remove meinprojekt.vigthoria.io
```

Hosting-Pläne anzeigen

```
# Verfügbare Hosting-Pläne und Preise anzeigen
vigthoria deploy plans
```

Befehlsreferenz

Kernbefehle

Befehl	Alias	Beschreibung
`vigthoria chat`	`vig c`	Interaktiven Chat starten
`vigthoria agent`	`vig a`	Agent-Modus starten
`vigthoria edit <datei>`	`vig e`	Datei mit KI bearbeiten
`vigthoria generate <beschr>`	`vig g`	Code generieren
`vigthoria explain <datei>`	`vig x`	Code erklären
`vigthoria fix <datei>`	`vig f`	Code-Probleme beheben
`vigthoria review <datei>`	`vig r`	Code-Qualität prüfen

Repository-Befehle

Befehl	Beschreibung
`vigthoria repo push`	Projekt in Vigthoria Repo hochladen
`vigthoria repo pull <name>`	Projekt aus Vigthoria Repo herunterladen
`vigthoria repo list`	Ihre Projekte auflisten
`vigthoria repo status`	Sync-Status prüfen
`vigthoria repo share <name>`	Teilbaren Link generieren
`vigthoria repo delete <name>`	Projekt aus Cloud entfernen
`vigthoria repo clone <url>`	Öffentliches Projekt klonen

Hub/Marktplatz-Befehle

Befehl	Beschreibung
`vigthoria hub discover`	Interaktive Modul-Entdeckung
`vigthoria hub list`	Alle verfügbaren Module auflisten
`vigthoria hub search <anfrage>`	Semantische Suche nach Modulen
`vigthoria hub info <modul>`	Modul-Details abrufen
`vigthoria hub activate <modul>`	Ein Modul aktivieren
`vigthoria hub active`	Ihre aktiven Module anzeigen

Bereitstellungs-Befehle

Befehl	Beschreibung
`vigthoria deploy preview`	Auf Vorschau-URL bereitstellen
`vigthoria deploy subdomain <name>`	Auf name.vigthoria.io bereitstellen
`vigthoria deploy custom <domain>`	Auf eigene Domain bereitstellen
`vigthoria deploy list`	Alle Bereitstellungen auflisten
`vigthoria deploy status [domain]`	Bereitstellungsstatus prüfen
`vigthoria deploy verify <domain>`	DNS-Konfiguration prüfen
`vigthoria deploy plans`	Hosting-Pläne anzeigen

Befehl	Beschreibung
`vigthoria deploy remove <domain>`	Bereitstellung entfernen

Auth-Befehle

Befehl	Beschreibung
`vigthoria login`	Bei Vigthoria authentifizieren
`vigthoria logout`	Authentifizierung löschen
`vigthoria status`	Kontostatus anzeigen

Konfig-Befehle

Befehl	Beschreibung
`vigthoria config`	Interaktive Konfiguration
`vigthoria config --list`	Alle Einstellungen anzeigen
`vigthoria config --set key=value`	Wert setzen
`vigthoria config --get key`	Wert abrufen
`vigthoria config --reset`	Auf Standardwerte zurücksetzen
`vigthoria init`	Projektkonfiguration initialisieren

Arbeitsabläufe & Beispiele

Arbeitsablauf 1: Feature-Entwicklung

```
# 1. Chat mit Projektkontext starten
cd ~/projekte/meine-app
vigthoria chat

# 2. Das Feature besprechen
you › Ich muss Benutzerauthentifizierung mit JWT hinzufügen

# 3. Den Code generieren
vigthoria generate "JWT-Authentifizierungs-Middleware für Express" -o src/auth/middleware.

# 4. Den generierten Code überprüfen
vigthoria review src/auth/middleware.ts

# 5. Eventuelle Probleme beheben
vigthoria fix src/auth/middleware.ts -t security
```

Arbeitsablauf 2: Code-Review vor dem Commit

```
# Pre-Commit-Hook erstellen
#!/bin/bash
# .git/hooks/pre-commit

for file in $(git diff --cached --name-only --diff-filter=ACM | grep '\.ts$'); do
```

```
  # Jede Datei überprüfen
  result=$(vigthoria review "$file" -f json)
  score=$(echo "$result" | jq '.score')

  if [ "$score" -lt 70 ]; then
    echo "Code-Review fehlgeschlagen für $file (Punktzahl: $score)"
    exit 1
  fi
done
```

Fehlerbehebung

Häufige Probleme

"Nicht authentifiziert" Fehler

```
# Lösung: Erneut anmelden
vigthoria login
```

"Abonnement abgelaufen" Warnung Ihr Abonnement muss erneuert werden. Besuchen Sie: https://coder.vigthoria.io/settings (Konto-Bereich)

"API-Verbindung fehlgeschlagen" Fehler

```
# API-Status prüfen
vigthoria status

# Alternativen Endpunkt versuchen
vigthoria config --set apiUrl=https://backup.coder.vigthoria.io
```

Alles zurücksetzen

```
# Konfiguration und Auth löschen
vigthoria logout
vigthoria config --reset

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

Sophie — Autonome Schleifen-Engine

Was ist Sophie?

Sophie ist Vigthorias **Autonome Schleifen-Engine** — ein persistentes KI-Iterationssystem, das eine Aufgabe entgegennimmt, sie in einer kontinuierlichen Schleife an KI-Modelle weiterleitet, bei jedem Schritt auf Fertigstellung prüft und so lange iteriert, bis die Aufgabe vollständig erledigt ist oder das maximale Iterationslimit erreicht wird.

Sophie ist Ihre unermüdliche KI-Partnerin: Planen → Ausführen → Bewerten → Verfeinern → Wiederholen — bis die Aufgabe wirklich fertig ist.

Schnellstart

```
# Einfache Aufgabe ausführen
sophie run "Erstelle eine REST-API mit Express.js und JWT-Authentifizierung"

# Mit Optionen ausführen
sophie run "Erstelle ein React-Dashboard mit Diagrammen" \
  --name "Dashboard-Projekt" \
  --max-iterations 25 \
  --strategy incremental \
  --workspace ./mein-projekt

# Aus einer Aufgabendatei ausführen
sophie run --file projektanforderungen.md --strategy test_driven
```

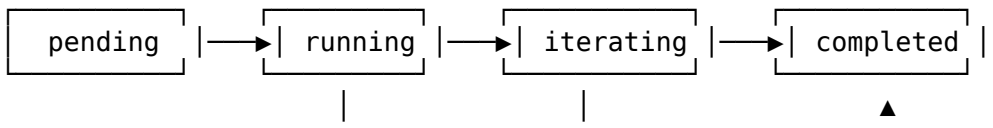
Sophie-Befehle

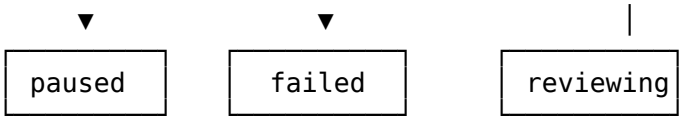
Befehl	Beschreibung
`sophie run "Aufgabe"`	Autonome Schleife erstellen und sofort starten
`sophie run --file <Pfad>`	Aufgabe aus Markdown/Text-Datei starten
`sophie status`	Engine-Status anzeigen (aktive Schleifen, Token-Verbrauch)
`sophie list`	Alle Sophie-Aufgaben auflisten
`sophie list --status running`	Aufgaben nach Status filtern
`sophie detail <task_id>`	Detaillierte Informationen zu einer Aufgabe
`sophie history <task_id>`	Vollständige Iterationshistorie anzeigen
`sophie pause <task_id>`	Laufende Schleife pausieren
`sophie resume <task_id>`	Pausierte Schleife fortsetzen
`sophie cancel <task_id>`	Schleife abbrechen und stoppen
`sophie start <task_id>`	Wartende/Entwurfs-Aufgabe starten
`sophie squad <args>`	Durchleitung — delegiert an Squad CLI

Iterationsstrategien

Strategie	Beschreibung	Ideal für
`persistent`	Iteriert mit vollem Kontext bis zum Abschlusssignal	Allgemeine Aufgaben
`test_driven`	Schreibt zuerst Tests, iteriert bis alle bestehen	Code mit klaren Testkriterien
`incremental`	Teilt Aufgabe in Phasen, vervollständigt jede nacheinander	Große Mehrstufen-Projekte
`review_refine`	Generieren, überprüfen, verfeinern in Zyklen	Qualitätskritische Lieferungen

Aufgabenlebenszyklus





KI-Backend (3-stufiges Fallback)

1. **Vigthoria Model Router** (localhost:4009) — intelligentes Modell-Routing
2. **OpenRouter** — Cloud-Fallback mit 100+ Modellen
3. **Ollama Lokal** — lokales Fallback für Offline/Private-Arbeit

---

Squad — Multi-Agenten-Teamsystem

Was ist Squad?

Squad ist Vigthorias **Multi-Agenten-Teamsystem** — ein Orchestrierungsframework, das spezialisierte KI-Agenten (Koordinator, Architekt, Programmierer, Tester usw.) einsetzt, die gemeinsam an großen Projekten arbeiten. Jeder Agent hat seine eigene Aufgabenwarteschlange, Terminal-Ausgabe und kommuniziert über einen gemeinsamen Nachrichtenbus.

Schnellstart

```
# Squad für ein Projekt erstellen
squad create "E-Commerce-Plattform mit Zahlungsintegration erstellen" \
  --agents coordinator,architect,coder,tester,docs

# Squad bereitstellen (startet die Ausführung)
squad deploy <squad_id>

# Fortschritt überwachen
squad status <squad_id>
squad agents <squad_id>
```

Squad-Befehle

Befehl	Beschreibung
`squad create "Mission"`	Neues Squad mit Missionsbeschreibung erstellen
`squad deploy <squad_id>`	Squad bereitstellen — startet die Agentenausführung
`squad status`	Übersicht aller Squads und der Engine
`squad status <squad_id>`	Detaillierter Status eines spezifischen Squads
`squad list`	Alle Squads auflisten
`squad agents <squad_id>`	Agenten im Squad mit Rollen und Status anzeigen
`squad terminal <sid> <aid>`	Terminal-Ausgabe eines Agenten anzeigen
`squad tasks <squad_id>`	Aufgabenaufschlüsselung des Squads anzeigen
`squad msg <sid> <aid> "msg"`	Direkte Nachricht an einen Agenten senden
`squad broadcast <sid> "msg"`	Nachricht an alle Agenten senden
`squad messages <squad_id>`	Nachrichtenprotokoll des Squads anzeigen
`squad pause <squad_id>`	Squad-Ausführung pausieren
`squad resume <squad_id>`	Pausiertes Squad fortsetzen

Befehl	Beschreibung
`squad disband <squad_id>`	Squad auflösen und abbrechen
`squad roles`	Alle verfügbaren Agentenrollen auflisten

Die 10 Agentenrollen

Rolle	Emoji	Spezialisierung
coordinator	🎯	Projekt in Aufgaben aufteilen, Arbeit zuweisen, Fortschritt überwachen
architect	🏗️	Systemarchitektur, APIs, Datenmodelle entwerfen
coder	💻	Features implementieren, Code schreiben
tester	🧪	Tests schreiben, Features validieren
debugger	🐛	Bugs diagnostizieren und beheben
docs	📚	Dokumentation, READMEs, API-Docs verfassen
reviewer	🔍	Code-Review, Standards durchsetzen
devops	🚀	Deployment, CI/CD-Pipelines, Infrastruktur
security	🔒	Sicherheitsaudits, Schwachstellenanalyse
data	📊	Datenbankdesign, Migrationen, Datenpipelines

Squad-Bereitstellungsablauf

```
1. 🎯 Koordinator analysiert die Mission
   └──▶ Erstellt JSON-Aufgabenplan mit Abhängigkeiten

2. Aufgaben werden Spezialagenten zugewiesen
   └──▶ Basierend auf Rollen & Abhängigkeitsreihenfolge

3. ⚡⚡⚡ Agenten führen Aufgaben parallel aus
   └──▶ Wo Abhängigkeiten es erlauben

4. 📡 Inter-Agenten-Kommunikation
   └──▶ Hilfeanfragen, Code-Reviews, Statusaktualisierungen

5. 📝 Datei-Schreiber V2
   └──▶ Agenten schreiben echte Dateien ins Projektverzeichnis

6. 🔗 Koordinator-Integrationsprüfung
   └──▶ Überprüft, ob alle Teile zusammenarbeiten

7. ✅ Squad abgeschlossen oder ❌ Fehlerbericht
```

Sophie + Squad zusammen

Sophie und Squad können zusammenarbeiten:

```
# Von Sophie an Squad delegieren
sophie squad create "Zahlungs-Microservice erstellen" --agents coder,tester,security
```

```
# Sophie für Einzelaufgaben, Squad für Multi-Agenten-Projekte
sophie run "Datenbankabfragen in /src/db/ optimieren" --strategy review_refine
squad create "Komplettes Frontend-Dashboard erstellen" --agents coordinator,coder,tester,d
```

FAQ

Allgemein

F: Ist Vigthoria CLI kostenlos? A: Grundfunktionen sind kostenlos. Erweiterte Modelle erfordern ein Pro- oder Ultra-Abonnement.

F: Funktioniert es offline? A: Nein, eine Internetverbindung ist für KI-Operationen erforderlich.

F: Wird mein Code in die Cloud gesendet? A: Ja, Code wird zur KI-Verarbeitung an Vigthoria-Server gesendet. Wir speichern Ihren Code nicht dauerhaft.

Technisch

F: Welche Modelle kann ich verwenden? A: Abhängig von Ihrem Abonnement. Prüfen Sie mit `vigthoria status`.

F: Kann ich es in CI/CD verwenden? A: Ja, verwenden Sie die Umgebungsvariable `VIGTHORIA_TOKEN` zur Authentifizierung.

Konto

F: Kann ich mein Vigthoria Coder-Konto verwenden? A: Ja, dasselbe Konto funktioniert für CLI, Web-IDE und VS Code-Erweiterung.

F: Wie kann ich mein Abonnement upgraden? A: Besuchen Sie https://coder.vigthoria.io/settings um Ihren Abonnementplan zu verwalten.

Hilfe erhalten

- **Dokumentation**: https://docs.vigthoria.io/cli
- **Discord-Community**: https://discord.gg/vigthoria
- **GitHub Issues**: https://github.com/vigthoria/vigthoria-cli/issues
- **E-Mail-Support**: support@vigthoria.io (Pro/Ultra-Abonnenten)

Viel Spaß beim Codieren mit Vigthoria CLI!

Mit ❤ erstellt von Vigthoria Technologie