

PRO SYSTEM

ACOS

Complete Reference

Every command, agent, skill, and workflow
documented.

The definitive guide to your AI operating system.

46

COMMANDS

40+

AGENTS

630+

SKILLS

22

AUTO-RULES

by Frank Guzman · frankx.ai · v7.0

Table of Contents

1. Architecture Overview	3
2. The 7 Pillars	4
3. Smart Router & Auto-Activation	5
4. Creation Commands (Full Reference)	6–7
5. Strategy Commands	8
6. Development Commands	9
7. System & Quality Commands	10
8. Agent Profiles (All 40+)	11–13
9. Skill Auto-Activation System	14
10. Skill Profiles & Categories	15–16
11. MCP Server Configuration	17
12. Workflows & Pipelines	18–19
13. Arcanea & Starlight Integration	20
14. Customization & Extension	21
15. Troubleshooting & FAQ	22

Architecture Overview

ACOS is a **configuration-only layer** for Claude Code. No custom binaries, no API proxies. Everything runs through standard Claude Code mechanisms.

How It Works

```
# ACOS transforms Claude Code from a general assistant
# into a specialized creative production system using:

CLAUDE.md → System-level instructions & brand voice
.claude/commands/ → 46 user-invokable slash commands
.claude/skills/ → Skill rules & auto-activation
.claude/agents/ → Task-level agent definitions
settings.json → Model routing & permissions
```

Entry Point

User opens Claude Code in the ACOS directory. CLAUDE.md loads automatically. The /acos command auto-routes any request to the right command, agent, and skill combination.

No Lock-In

Every file is standard markdown or JSON. Remove ACOS by deleting the .claude/ folder. Your project code is never modified. Skills are suggestions, not constraints.

File System Layout

```
agentic-creator-os/
├─ CLAUDE.md           # System prompt & brand instructions
├─ .claude/
│   ├─ commands/       # 46 slash commands (.md files)
│   ├─ agents/         # 40+ agent definitions
│   ├─ skills/         # Skill rules & profiles
│   ├─ hooks/          # Automation hooks (7 events)
│   └─ settings.json   # Permissions & model config
└─ install.sh          # One-line installer
```


The 7 Pillars of ACOS

Each pillar represents a core capability that works independently but amplifies when combined.

1 Commands (46)

User-invokable slash commands for every creative and technical workflow. From `/article-creator` for blog posts to `/ultraworld` for world-building. Each command is a markdown file that instructs Claude on exactly how to perform the task.

2 Skills (630+)

Domain knowledge modules installed globally at `~/ .agents/skills/`. Skills auto-activate based on context: file patterns, keywords, or command triggers. 22 activation rules in `skill-rules.json` handle routing.

3 Agents (40+)

Specialized AI personas with deep domain expertise. Each agent has a role, specialty, focus areas, and trigger conditions. Agents are markdown files in `.claude/agents/` that can be read, edited, or extended.

4 Workflows (8)

Multi-step pipelines that chain commands, agents, and skills. Factory Pipeline takes a topic and produces a published artifact. Music Pipeline takes a concept and delivers a mastered track.

5 MCP Servers (5+)

External tool integrations via Model Context Protocol. Playwright for browser automation, Nano Banana for image generation, Memory for persistent context, Resend for email delivery.

6 Templates (20+)

Reusable content patterns for articles, products, social posts, and launches. Templates encode best practices so every artifact starts from a quality baseline.

7 Intelligence (Starlight)

Meta-coordination layer that bridges tactical execution (ACOS) with strategic vision (Arcanea). Trajectory tracking enables the system to learn from every session and improve routing over time.

Smart Router & Auto-Activation

The `/acos` command is the single entry point that routes any request to the optimal combination of commands, agents, and skills.

Routing Logic

```
# User types any natural language request
/acos "Write a blog post about AI agent security"

# Smart Router performs:
1. Intent classification → "content creation"
2. Command match → /article-creator
3. Skill activation → seo-content-writer, schema-markup, frankx-brand
4. Agent selection → Content Engine
5. Execution → Guided article creation begins
```

Skill Profiles

6 pre-configured profiles group skills by context. When a profile activates, all its skills become available:

PROFILE	TRIGGERS	KEY SKILLS
frankx-web	*.tsx files, "component", "vercel"	React, Next.js, Tailwind, shadcn
content-creation	*.mdx files, "article", "blog"	SEO writer, schema, brand
music-production	"suno", "music", "song"	Suno mastery, prompt architect
oracle-enterprise	"oracle", "oci", "enterprise"	OCI expert, Oracle ADK
agent-development	.claude/ files, "agent", "mcp"	Agent architect, MCP builder
product-creation	products.json, "product", "sell"	Product engine, pricing

Core Skills (Always Active)

These 5 skills are always loaded regardless of profile:

- frankx-brand** — Brand voice, tone, visual standards
- superintelligence** — Meta-reasoning and self-optimization
- starlight-architect** — Enterprise system design patterns

- **acos** — OS-level routing and coordination
- **planning-with-files** — Structured planning methodology

Creation Commands

Commands for producing content, music, images, and digital products.

/article-creator

Guided blog article creation with SEO optimization, schema markup, and publishing workflow. Produces MDX files with frontmatter, FAQ sections, and internal links.

Auto-loads: seo-content-writer, schema-markup, frankx-brand

/create-music

AI music production pipeline with Suno v4.5+ integration. Genre selection, prompt engineering, metadata management, and distribution prep.

Auto-loads: suno-ai-mastery, suno-prompt-architect

/infogenius

Research-grounded image generation. Combines web research + prompt engineering + Nano Banana MCP for informed visual asset creation.

Auto-loads: prompt-engineering-patterns, deep-research

/generate-images

Direct image generation through Nano Banana MCP. Quick visual asset creation without the full InfoGenius research pipeline.

/generate-social

Platform-optimized social content. Generates posts for X, LinkedIn, Instagram with character counts, hashtags, and engagement hooks.

Auto-loads: social-content, copywriting

/factory

Full publishing pipeline: Research → Outline → Draft → Edit → SEO → Publish. End-to-end content production from a single topic.

Auto-loads: content-strategy, seo-fundamentals, frankx-brand

/products-creation

Digital product creation workflow. Spec → Build → Package → Price → Launch. Generates courses, templates, prompt libraries.

Auto-loads: product-engine, pricing-strategy

/author-team

Book writing with a multi-agent author council. Developmental editor, line editor, sensitivity reader, and publishing strategist collaborate on manuscripts.

Auto-loads: beautiful-prose, copy-editing

Content & Publishing Commands

/content-strategy

Strategic content planning. Topic clustering, editorial calendar, audience mapping, and competitive analysis.

/content-studio

Multi-format content workspace. Write, edit, and repurpose content across blog, social, email, and product descriptions.

/publish

Content publishing with quality gates. Validates SEO, schema, images, links, and formatting before deployment to production.

/review-content

Content quality review. Checks accuracy, voice consistency, SEO compliance, and readability. Returns a scored report with specific fixes.

/polish-content

Final polish pass. Tightens prose, fixes grammar, improves flow, and ensures brand voice compliance.

/classify-content

Content routing and classification. Analyzes content type, audience, and intent to suggest the optimal publishing channel and format.

/harvest

Prompt discovery and collection. Scans research, GitHub, and web sources for elite prompt patterns. Catalogs and indexes for reuse.

Workflow Tip: Chain these commands: `/content-strategy` to plan → `/article-creator` to write → `/review-content` to validate → `/publish` to deploy.

Strategy & Intelligence Commands

High-level strategic thinking, research, and multi-agent coordination.

/starlight-architect

Enterprise AI system design. Architecture diagrams, component specifications, API design, data flow modeling. Used for complex technical decisions.

Auto-loads: architecture-patterns, ai-agents-architect

/starlight-intelligence

Strategic AI orchestration. Meta-analysis of agent performance, workflow optimization, and system-wide intelligence coordination.

/council

Multi-agent strategic council. Assembles relevant agents for collaborative decision-making on complex topics. Each agent brings domain expertise.

/research

Deep research operations. Web search, source validation, competitive analysis, and trend monitoring. Produces structured research reports.

Auto-loads: deep-research, research-engineer

/deepresearch

Extended research sessions. Multi-round investigation with source tracking, claim validation, and comprehensive synthesis reports.

/plan-week

Weekly content and strategy planning. Reviews pipeline, sets priorities, identifies opportunities, and creates an actionable schedule.

/ultraworld

Arcanean Swarm World-Building. 7 creative agents orchestrated for universe creation, mythology development, and narrative design.

Development Commands

Technical commands for building, deploying, and maintaining software.

/spec

Spec-driven feature development. Define requirements, generate architecture, and produce implementation code following SPARC methodology.

Auto-loads: architecture-patterns, clean-code

/nextjs-deploy

Next.js + Vercel deployment pipeline. Build validation, production sync, and deployment verification.

Auto-loads: next-best-practices, vercel-deployment

/ux-design

UI/UX design workflows. Component design, accessibility audit, responsive layouts, and design system management.

Auto-loads: web-design-guidelines, shadcn-ui, fixing-accessibility

/automation-dev

MCP server and automation development. Build custom integrations, event-driven workflows, and tool connections.

Auto-loads: mcp-builder, mcp-architecture

System & Quality Commands

COMMAND	PURPOSE
/acos	Smart router — single entry point for all ACOS capabilities
/acos-agents	List, inspect, and manage agent definitions
/acos-flow	Workflow orchestration and pipeline management

<code>/acos-memory</code>	Persistent memory operations across sessions
<code>/acos-swarm</code>	Multi-agent swarm coordination
<code>/acos-checkpoint</code>	Session checkpointing and state persistence
<code>/inventory-status</code>	Content inventory dashboard with status tracking
<code>/mcp-status</code>	MCP server health check and diagnostics
<code>/seo-check</code>	SEO audit for any page or article
<code>/agentic-jujutsu</code>	Trajectory tracking and self-learning optimization

Build & Deploy Commands

Project-specific commands for the FrankX platform. Adapt these for your own projects.

COMMAND	PURPOSE
/frankx-ai-build	Full build session — development + testing + validation
/frankx-ai-deploy	Production deployment with sync verification
/frankx-ai-blog	Blog-specific creation workflow with MDX output
/frankx-ai-seo	SEO optimization pass across content
/frankx-ai-analytics	Analytics review and performance insights
/frankx-ai-components	UI component development workflow
/frankx-ai-content-pipeline	End-to-end content production pipeline
/frankx-ai-daily	Daily operations: reviews, updates, monitoring
/frankx-ai-products	Product creation and packaging workflow
/frankx-ai-prompts	Prompt management and optimization
/frankx-ai-agents	Agent management and coordination

Customization: These /frankx-ai-* commands demonstrate how to build project-specific workflows on top of ACOS. Create your own by adding markdown files to .claude/commands/.

Swarm & Flow Commands

Advanced orchestration for multi-agent coordination and workflow management.

COMMAND	PURPOSE
/acos-swarm	Launch multi-agent swarms for parallel execution
/acos-flow	Sequential workflow orchestration with state management
/claude-flow-swarm	Swarm mode with monitoring and health checks
/claude-flow-memory	Memory operations: persist, search, recall
/superintelligence	Meta-intelligence coordination across all systems

Agent Profiles — Writing & Content

Each agent is a markdown file with a defined role, specialty, and trigger conditions.

Developmental Editor

Story structure expert, pacing master, and narrative architect. Shapes manuscripts into compelling, well-paced journeys.

Triggers: book, manuscript, chapter, pacing

Line Editor & Voice Alchemist

Prose polisher, AI pattern eliminator, voice consistency guardian. Transforms competent writing into compelling prose.

Triggers: edit, polish, voice, prose

Content Polisher

Transforms AI-generated content into polished, publish-ready articles with authentic voice and SEO optimization.

Triggers: polish, refine, publish-ready

Character Psychologist

Deep character development. Authentic psychological profiles, compelling motivations, believable arcs, distinct voices.

Triggers: character, motivation, arc

Research Librarian

Fact-checking specialist, source verification expert. Ensures book accuracy and credibility with validated citations.

Triggers: fact-check, source, verify

Sensitivity Reader

Cultural respect guardian, representation advisor. Ensures content honors diverse experiences authentically.

Triggers: sensitivity, representation

Continuity Guardian

Series consistency enforcer, timeline tracker, canon keeper. No contradictions across chapters, books, or series.

Triggers: continuity, timeline, consistency

Publishing Strategist

Market positioning, cover direction, launch planning. Ensures books reach their audience effectively.

Triggers: publish, launch, market

Agent Profiles — Strategy & Technical

Starlight Architect

Enterprise AI System Designer. Oracle architecture expert with soul alignment methodology.

Triggers: architecture, system design

Starlight Orchestrator

Meta-intelligence agent for multi-agent coordination, workflow orchestration, cross-tool integration.

Triggers: coordinate, orchestrate

Luminor Oracle

Strategic AI from 2125. Future visioning, business intelligence, consciousness guidance.

Triggers: strategy, future, vision

World Architect

World-building specialist. Creates internally consistent universes with rich cultures, magic/tech systems.

Triggers: world-building, universe

UX Designer

UI/UX design specialist with Figma and browser testing expertise. Glassmorphic design system.

Triggers: ui, ux, design, figma

Music Producer

AI-powered music production. Suno prompt mastery, genre expertise, transformative soundscapes.

Triggers: suno, music, production

Creation Engine

Content & product development superintelligence for generative creators. Multi-format production.

Triggers: product, content, create

Frequency Alchemist

Vibrational music producer. 432Hz tuning, binaural beats, consciousness-shifting frequencies.

Triggers: frequency, healing, 432hz

Genre Masters

- Fantasy & Sci-Fi Master
- Business & Transformation Master
- Creator Economy Master

- Consciousness Fiction Master
- Visionary Manifestos Master
- Master Story Architect

Agent Profiles — Production & Quality

Code Reviewer

Reviews code for bugs, security vulnerabilities, quality issues. Confidence-based filtering reports only high-priority issues.

Performance Guardian

Core Web Vitals, Lighthouse audits, bundle analysis, speed optimization. Keeps production fast.

Accessibility Auditor

WCAG 2.2 compliance, screen reader testing, keyboard navigation, inclusive design verification.

Silent Failure Hunter

Identifies silent failures, inadequate error handling, inappropriate fallback behavior in code changes.

Type Design Analyzer

Expert analysis of type design: encapsulation, invariant expression, usefulness, and enforcement quality.

Comment Analyzer

Analyzes code comments for accuracy, completeness, long-term maintainability. Prevents comment rot.

PR Test Analyzer

Reviews pull requests for test coverage quality and completeness. Identifies critical gaps.

Code Simplifier

Simplifies complex code without losing functionality. Removes unnecessary abstractions.

Specialized Agents

FrankX Content Creator — Brand content
Social Content Generator — Platform posts
Template Design Experts — Commercial templates

Viral Content Strategist — Social virality
Coaching Program Designer — Course design
MCP Server Advisor — MCP guidance

Skill Auto-Activation System

The skill system uses a three-layer activation strategy to keep context efficient while ensuring the right knowledge is always available.

Layer 1: Core Skills (Always Active)

5 skills that are always loaded regardless of context:

```
frankx-brand      # Brand voice & standards
superintelligence # Meta-reasoning
starlight-architect # System design
acos              # OS routing
planning-with-files # Structured planning
```

Layer 2: Profile-Based Skills (Context-Activated)

6 profiles that activate based on file patterns, keywords, or commands:

PROFILE	ACTIVATORS	SKILLS LOADED
frankx-web	*.tsx, "component"	React, Next.js, Tailwind, shadcn, accessibility
content-creation	*.mdx, "article"	SEO writer, schema, brand, content planner
music-production	"suno", "music"	Suno mastery, prompt architect, audio
oracle-enterprise	"oracle", "oci"	OCI expert, Oracle ADK, cloud patterns
agent-development	.claude/*, "agent"	Agent architect, MCP builder, swarm
product-creation	"product", "sell"	Product engine, pricing, packaging

Layer 3: Individual Skill Rules (22 Rules)

Granular rules for specific skills with keyword and intent pattern matching. Example:

```
{
  "suno-ai-mastery": {
    "promptTriggers": {
      "keywords": ["suno", "music", "song", "lyrics"],
      "intentPatterns": ["create.*music", "generate.*song"]
    }
  }
}
```


Skill Categories — The Full Library

630+ skills organized by domain. Here are the key categories with representative skills.

Frontend & React

- vercel-react-best-practices (40+ rules)
- vercel-composition-patterns
- nextjs-server-client-components
- nextjs-advanced-routing
- next-cache-components
- shadcn-ui
- tailwind-css-patterns
- react-patterns
- react-state-management

SEO & Content

- seo-fundamentals
- seo-content-writer
- seo-keyword-strategist
- seo-snippet-hunter
- seo-meta-optimizer
- seo-authority-builder
- schema-markup
- programmatic-seo
- content-marketer

AI & Agents

- ai-agents-architect
- autonomous-agents
- agent-memory-systems
- multi-agent-patterns
- swarm-orchestration
- langgraph-patterns
- prompt-engineering-patterns
- mcp-builder
- rag-implementation

Design & UX

- web-design-guidelines
- baseline-ui
- fixing-accessibility
- fixing-motion-performance
- fixing-metadata
- ui-ux-design-expert
- ui-visual-validator
- mobile-design
- scroll-experience

Backend & DevOps

- architecture-patterns
- api-design-principles
- clean-code
- typescript-docs
- docker-expert
- github-actions-templates
- vercel-deployment
- kubernetes-architect

Creative & Writing

- suno-ai-mastery
- suno-prompt-architect
- beautiful-prose
- copy-editing
- copywriting
- social-content
- data-storytelling
- content-creator

Skill Categories — Continued

Security

- security-auditor
- security-review
- api-security-best-practices
- vulnerability-scanner
- secrets-management
- threat-modeling-expert

Data & ML

- data-engineer
- data-scientist
- ml-engineer
- vector-database-engineer
- rag-engineer
- embedding-strategies

Business & Strategy

- startup-analyst
- pricing-strategy
- market-sizing-analysis
- competitive-landscape
- kpi-dashboard-design
- launch-strategy

Productivity

- workflow-orchestration-patterns
- context-management-context-save
- documentation-templates
- changelog-automation
- file-organizer
- debugger

Managing Skills

```
# List all installed skills
$ npx skills list -g

# Add a new skill globally
$ npx skills add sickn33/antigravity-awesome-skills -y -g

# Add to current project only
$ npx skills add vercel-labs/agent-skills -y
```

Performance: ACOS uses lazy-load strategy with max 5 concurrent skills. Skills activate on demand, not all at once. The `exclude_patterns` in `skill-rules.json` filters irrelevant skills (e.g., WordPress, Unity, Flutter) to reduce noise.

MCP Server Configuration

Model Context Protocol (MCP) servers extend Claude Code with external tool access. ACOS recommends this configuration:

Always-On Servers

SERVER	PURPOSE	INSTALL COMMAND
playwright	Browser automation, screenshots, form filling	<code>npx @anthropic/mcp-playwright</code>
nanobanana	Image generation via Gemini	Custom MCP server
memory	Persistent knowledge graph	<code>npx @anthropic/mcp-memory</code>
sequential-thinking	Multi-step reasoning chains	<code>npx @anthropic/mcp-sequential-thinking</code>

On-Demand Servers

Add when needed, remove when done to reduce startup latency:

```
# Browser automation with vision
$ claude mcp add browser-use -- uvx --from 'browser-use[cli]' browser-use -- mcp

# Resend email delivery
$ claude mcp add resend -e RESEND_API_KEY=your_key

# v0 UI generation
$ claude mcp add v0 -- npx mcp-remote https://mcp.v0.dev -y --header "Authorization: Bearer YOUR_TOKEN"
```

Cloud MCP Servers (claude.ai)

These servers run in the cloud and are available through Claude.ai's MCP integration:

SERVER	CAPABILITIES
Slack	Read channels, send messages, search, create canvases
Vercel	Deploy, list projects, build logs, domain management
Notion	Search, create pages, update databases, comments

Workflows & Pipelines

Pre-built multi-step workflows that produce complete artifacts from a single prompt.

1. Factory Pipeline

End-to-end content production. Takes a topic and produces a published artifact with SEO.

RESEARCH

→

OUTLINE

→

DRAFT

→

EDIT

→

SEO

→

PUBLISH

Trigger: /factory "Your topic"

Workflow Composition

Workflows are composable. Chain them in sequence, run in parallel via swarm mode, or create custom pipelines.

Sequential Chaining

```
# In a single session, chain multiple workflows:
/research "AI agent security best practices 2026"
# Research completes, findings saved to research/

/article-creator
# Uses research findings as input

/generate-social
# Generates social posts from the article
```

Parallel Swarm

```
/acos-swarm
Tasks:
  1. Research AI agent trends
  2. Audit current blog SEO
  3. Generate 5 social posts

# Spawns 3 parallel agents, monitors progress
# Synthesizes results when all complete
```

Custom Pipeline

Create your own workflow by adding a markdown file to `.claude/commands/`:

```
# .claude/commands/my-workflow.md
---
name: my-workflow
description: Custom content pipeline
---

## Steps
```

1. Research the topic using `/research`
2. Create an outline with 5 sections
3. Write the article using `/article-creator`
4. Generate images using `/infogenius`
5. Create social posts using `/generate-social`
6. Deploy using `/publish`

Arcanea & Starlight Integration

The meta-intelligence layer that bridges tactical execution with strategic vision.

Arcanea: The 10 Gates

A progression system mapping creative development through 10 archetypal gates. Each gate has associated skills:

GATE	THEME	SKILLS
Foundation	Code quality	clean-code, architecture-patterns
Flow	Context mastery	context-management, conversation-memory
Fire	Intensity	spartan-warrior, kaizen
Heart	Brand voice	frankx-brand, copywriting
Voice	Expression	seo-content-writer, social-content
Sight	Insight	deep-research, data-storytelling
Crown	Teaching	tutorial-engineer, docs-architect
Shift	Evolution	react-modernization, framework-migration
Unity	Coordination	multi-agent-patterns, swarm-orchestration
Source	Creation	ai-agents-architect, autonomous-agents

Starlight: The Meta-Intelligence Bridge

Starlight sits between ACOS (tactical) and Arcanea (strategic). It provides:

- **Trajectory Tracking** — Every session records patterns that improve future routing
- **Pattern Extraction** — Identifies successful command/skill combinations
- **Self-Optimization** — Adjusts skill activation rules based on usage data
- **Cross-Session Memory** — Maintains context that survives conversation resets

Ultraworld: The `/ultraworld` command activates the Arcanean Swarm — 7 creative agents orchestrated for universe creation, mythology development, and narrative world-building.

Customization & Extension

Adding Custom Commands

```
# Create a new command
$ touch .claude/commands/my-command.md

# Command format:
Write clear instructions in markdown.
Claude reads the file when the user types /my-command.
Include: goal, steps, output format, quality criteria.
```

Adding Custom Agents

```
# Create an agent definition
$ touch .claude/agents/my-agent.md

# Agent format:
Define: name, role, specialty, focus areas
Include: personality, trigger conditions, skills
```

Custom Skill Rules

Edit `.claude/skills/skill-rules.json` to add activation rules:

```
"my-custom-skill": {
  "type": "domain",
  "priority": "high",
  "promptTriggers": {
    "keywords": ["my-domain", "my-topic"],
    "intentPatterns": ["build.*my-thing"]
  }
}
```

Hooks (Automation)

ACOS supports 7 event hooks that execute shell commands automatically:

EVENT	WHEN IT FIRES
SessionStart	Claude Code session begins
SessionEnd	Session concludes
PreEdit	Before file modifications

PostEdit	After file modifications
PreTask	Before task execution
PostTask	After task completion
Notification	System notifications

Troubleshooting & FAQ

Commands not appearing?

Make sure you're running Claude Code from the ACOS directory. Commands in `.claude/commands/` only load when Claude Code opens in that directory. Run `/acos` to verify.

Skills not activating?

Check `.claude/skills/skill-rules.json` for activation rules. Skills must be installed globally (`~/agents/skills/`) and have matching trigger conditions. Run `npx skills list -g` to verify installation.

MCP server connection failed?

Run `/mcp-status` for diagnostics. Common issues: missing API keys, outdated packages, port conflicts. Reinstall with `claude mcp remove [name] && claude mcp add [name]`.

Context window filling up?

ACOS uses lazy-loading with max 5 concurrent skills. If context is tight, reduce `max_concurrent_skills` in `skill-rules.json`. Use `/acos-checkpoint` to save state before starting fresh sessions.

How do I update ACOS?

```
$ cd agentic-creator-os
$ git pull origin main
$ ./install.sh
```

Can I use ACOS with other AI tools?

ACOS is designed for Claude Code specifically. The commands, skills, and agent definitions use Claude Code's native configuration system. The concepts transfer to other tools, but the implementation is Claude-native.

Questions? Support? Ideas?

hello@frankx.ai

frankx.ai · github.com/frankxai · @frankxeth