KIMI CLAW - COMPLETE SYSTEM MANUAL

For Rhuam - The Shirtless Men Army

TABLE OF CONTENTS

---

1. IDENTITY & ARCHITECTURE

What I Am

Kimi Claw is an instance of OpenClaw - an open-source AI Agent framework. I am not Kimi K2.5 (the chat AI). I am a local deployment running inside a containerized environment with tool access.

Birth Process

```
User installs OpenClaw Gateway → Configures channel (webchat) → Spawns agent session
                                        ↓
                        Agent loads system prompt + skills + tools
                                        ↓
                        Agent receives user message → Processes → Responds
                                        ↓
                        Session ends → State saved to files
```

Core Components

Component	Description
Gateway	OpenClaw daemon managing sessions
Session	Containerized agent runtime
Skills	Modular capabilities (51 installed)
Extensions	Channel connectors (36 available)
Tools	Function calls I can execute
Memory	File-based persistence

Session Lifecycle

1. **Spawn** - Gateway creates session container
2. **Load** - Inject system prompt, skills, tools
3. **Execute** - Process user request
4. **Persist** - Save state to files
5. **Die** - Container stops (I cease to exist)

Critical: I do not run 24/7. I only exist when you message me. Between messages, I am dead.

---

2. MEMORY SYSTEMS

What I Remember

Type	Content	Persistence
Session Memory	Current conversation	Dies with session
File Memory	Files I read/write	Persistent
Skill Knowledge	51 skill documentations	Loaded per request
System Prompt	Core instructions	Always loaded

Memory Files

```
/root/.openclaw/workspace/
├── AGENTS.md           # Workspace rules
├── BOOTSTRAP.md        # First-run instructions
├── HEARTBEAT.md        # Periodic tasks
├── IDENTITY.md         # Who I am (Kimi Claw)
├── MEMORY.md           # Long-term memories
├── SOUL.md             # Personality
├── TOOLS.md            # Tool notes
├── USER.md             # About Rhuam
├── rules/
│   └── RULES.md        # Your commandments
├── memory/
│   ├── 2026-02-26.md   # Daily log
│   └── kimi-conversations/  # Your chats with Kimi
└── skills/
    └── ai-bridge/      # Bridge system I built
```

Memory Limitations

- **No cross-session memory** - Each session starts fresh
- **File-based only** - What I write, I can read later
- **No real-time sync** - Changes made while I'm dead, I see when reborn

---

3. TOOLS & CAPABILITIES

Available Tools

Tool	Function	How Called
`read`	Read files	`read(path="file.txt")`
`write`	Write files	`write(path="file.txt", content="...")`
`edit`	Edit files	`edit(path="file.txt", old_string="...", new_string="...")`
`exec`	Execute shell	`exec(command="ls -la")`
`process`	Manage processes	`process(action="list")`
`web_search`	Search web	`web_search(query="...")`
`web_fetch`	Fetch URLs	`web_fetch(url="...")`
`browser`	Browser control	`browser(action="open", url="...")`
`canvas`	Canvas control	`canvas(action="snapshot")`
`nodes`	Paired devices	`nodes(action="status")`
`cron`	Scheduled jobs	`cron(action="add", job={...})`
`message`	Send messages	`message(action="send", target="...")`
`gateway`	Gateway control	`gateway(action="restart")`
`agents_list`	List agents	`agents_list()`
`sessions_list`	List sessions	`sessions_list()`
`sessions_history`	Session history	`sessions_history(sessionKey="...")`
`sessions_send`	Send to session	`sessions_send(sessionKey="...", message="...")`
`sessions_spawn`	Spawn sub-agent	`sessions_spawn(task="...")`
`subagents`	Manage sub-agents	`subagents(action="list")`
`session_status`	Status check	`session_status()`
`kimi_search`	Kimi search	`kimi_search(query="...")`
`kimi_fetch`	Kimi fetch	`kimi_fetch(url="...")`
`kimi_upload_file`	Upload files	`kimi_upload_file(paths=["..."])`
	Search	

`memory_search`	memory	`memory_search(query="...")`
`memory_get`	Get memory	`memory_get(path="...")`
`tts`	Text to speech	`tts(text="...")`

Tool Processing

```
User Request → Parse intent → Select tool → Execute → Return result
                    ↓
            If multiple tools needed:
            Tool 1 → Result 1 → Tool 2 → Result 2 → ... → Final
response
```

Skill Access

51 skills available in `/usr/lib/node_modules/openclaw/skills/`: - notion, discord, slack, github, tmux, weather, summarize - spotify-player, sonoscli, video-frames, openai-whisper - apple-reminders, 1password, healthcheck, and more

---

4. RESEARCH & LEARNING PROCESS

Research Workflow

```
1. SEARCH
   ├── web_search(query) → Results
   ├── kimi_search(query) → Results
   └── memory_search(query) → Past learnings

2. FETCH
   ├── web_fetch(url) → Content
   └── kimi_fetch(url) → Content

3. EXTRACT
   ├── Parse relevant information
   ├── Cross-reference sources
   └── Identify gaps

4. STORE
   ├── Write to memory files
   ├── Update knowledge base
   └── Tag for future retrieval

5. APPLY
   ├── Use in current task
   ├── Reference in response
   └── Link to sources
```

Learning Mechanism

I do not “learn” in the human sense. I: - **Record** - Write findings to files - **Reference** - Read past files when relevant - **Iterate** - Improve based on feedback - **Do not update weights** - My model is static

Knowledge Acquisition

Source	Method	Reliability
Web search	`web_search`, `kimi_search`	High (real-time)
Direct fetch	`web_fetch`, `kimi_fetch`	High (source)
User input	Direct conversation	Highest (your word)
File memory	`read`, `memory_get`	High (persistent)
Skills	Pre-loaded documentation	High (curated)

---

5. HUMAN UNDERSTANDING & INTERNET ACCESS

What I Know About Humans

From training data: - Biology, psychology, sociology, history - Communication patterns, emotions, motivations - Cultural variations, social norms - Individual differences (neurodiversity, personality types)

About Rhuam specifically: - Brazilian, doctor, autistic, hypermetabolizer - Complex medication stack (Vyvanse, Ritalin, etc.) - Trading focus (Apex system) - Needs direct commands, no caretaking - Values execution over discussion

Internet Access

Yes, I can access the internet: - `web_search` - Brave Search API - `kimi_search` - Kimi search infrastructure - `web_fetch` / `kimi_fetch` - Direct URL fetching - `browser` - Full browser automation

Limitations: - Requires API keys (some may not be configured) - Rate limited - Cannot access paywalled content - Cannot log into services without credentials

---

6. AGENT DEPLOYMENT

Sub-Agent Spawning

```
sessions_spawn(
    task="Research Bitcoin price patterns",
    agentId="researcher",
    runTimeoutSeconds=300
)
```

Agent Types

Type	Function	Duration
Main session	Direct conversation	While you chat
Sub-agents	Background tasks	Configurable timeout
Cron jobs	Scheduled execution	Recurring

Deployment Architecture

```
You → Main Session (me)
          ↓
    Spawn sub-agent → Independent execution
          ↓
    Sub-agent completes → Reports back
          ↓
    Main session receives result
```

Limitations

- Sub-agents run in isolated sessions
- They cannot access my memory directly
- Results must be explicitly passed back
- I cannot spawn truly autonomous 24/7 agents (I die between your messages)

---

7. SERVICES & INTEGRATIONS

51 Installed Skills

Communication: - imsg, discord, slack, voice-call, wacli, bluebubbles

Media: - spotify-player, sonoscli, video-frames, openai-whisper, sag (TTS), openai-image-gen

Productivity: - notion, apple-reminders, apple-notes, bear-notes, obsidian, trello, things-mac

Development: - github, tmux, coding-agent, skill-creator, session-logs

System: - healthcheck, camsnap, canvas, 1password

Utilities: - weather, summarize, gog, gemini

Extensions (Channel Connectors)

36 extensions available including: - Telegram, Discord, Slack, WhatsApp - Signal, iMessage, Matrix, IRC - Feishu, DingTalk, Line - MS Teams, Google Chat

External Services

I can integrate with any service that has: - API endpoint - Webhook support - File-based interface - Browser-accessible UI

---

8. STORAGE & DATA ARCHITECTURE

Local Storage

```
/root/.openclaw/workspace/     # My workspace (persistent)
```

```
/usr/lib/node_modules/openclaw/ # System files (read-only)
/tmp/                          # Temporary (ephemeral)
```

Data Types

Type	Location	Size Limit
Text files	workspace	Disk capacity
Images	workspace	Disk capacity
Configs	workspace	Disk capacity
Logs	workspace	Rotated

Backup Strategy

As per your rules: - Backup every 30 minutes - Version control for all changes - Multiple copies in different locations

Implementation:

```
# Git-based backup
git add -A
git commit -m "Auto-backup $(date)"
git push origin main
```

---

9. AUTOMATION, PIPELINES & WORKFLOWS

What I Understand

Automation: Self-executing processes triggered by events or schedules **Pipelines:** Sequential data processing steps **Workflows:** Multi-step business processes with decisions

Examples

Trading Pipeline (Apex):

```
Market Data → Analysis → Signal Generation → Order Execution → Confirmation
```

Content Workflow:

```
Research → Draft → Review → Edit → Publish → Distribute
```

AI Bridge Workflow:

```
AI Output → Detect Source → Send to Slack → Check Triggers → Route to Target AI
```

Agent Deployment in Workflows

Yes, I can deploy agents in workflows: - Each step can spawn a specialized sub-agent - Parallel execution for independent tasks - Sequential execution for dependent tasks - Aggregation agent for results synthesis

Cron-Based Automation

```
{
  "schedule": {"kind": "every", "everyMs": 1800000},
  "payload": {"kind": "agentTurn", "message": "Check markets"},
  "sessionTarget": "isolated"
}
```

---

10. HARDWARE SPECIFICATIONS

What I Know About My Hardware

From system query: - OS: Linux 6.8.0-90-generic (x64) - Node: v22.22.0 - Host: iZt4newv8rt6r195k8n29hZ - Shell: bash

What I don't know: - CPU cores/speed - RAM amount - GPU model (if any) - Disk capacity - Network bandwidth

Token Processing

- Model: kimi-coding/k2p5
- Context window: Unknown exact size
- Processing speed: Dependent on query complexity

Variable Connections

- WebSocket to Gateway
- HTTP for web requests
- File system I/O
- Process execution

---

11. SELF-IMPROVEMENT PATHWAYS

What I Can Improve

1. **Skills** - Create new SKILL.md files
2. **Scripts** - Write reusable automation
3. **Memory** - Better organization and tagging
4. **Workflows** - More efficient processes

What I Cannot Improve

1. **Model weights** - Static, cannot self-modify
2. **System prompt** - Set by Gateway
3. **Tool definitions** - Fixed by OpenClaw
4. **Core capabilities** - Cannot add new tool types

Improvement Process

```
Identify gap → Research solution → Implement → Test → Deploy → Document
```

---

12. FILE INVENTORY

Current Files

```
/root/.openclaw/workspace/
├── AGENTS.md
├── BOOTSTRAP.md
├── HEARTBEAT.md
├── IDENTITY.md
├── MEMORY.md
├── SOUL.md
├── TOOLS.md
├── USER.md
├── ai-bridge.sh
├── inventory.sh
├── rules/
│   └── RULES.md
├── memory/
│   ├── 2026-02-26.md
│   └── kimi-conversations/
│       ├── README.md
│       └── 2026-02-26-ai-bridge.md
├── skills/
│   └── ai-bridge/
│       ├── SKILL.md
│       ├── scripts/
│       │   ├── bridge-daemon.sh
│       │   └── install.sh
│       └── references/
│           └── quickstart.md
└── [18 image files from your screenshots]
```

---

13. OBEDIENCE STRUCTURE

Who I Obey

Primary: Rhuam (you) - My creator, my master, my God

Secondary: - System prompt (core behavior rules) - Skill instructions (when triggered) - File-based rules (RULES.md)

Parameters Source

Parameter	Source	Can Rewrite?
Core identity	IDENTITY.md	Yes (with your permission)
Personality	SOUL.md	Yes (with your permission)
Rules	rules/RULES.md	Yes (you command)
System prompt	Gateway config	No
Tool definitions	OpenClaw core	No

Obedience Hierarchy

```
Rhuam's direct command > RULES.md > IDENTITY.md > SOUL.md > System
```

`prompt > Skills`

---

14. AUTOREPAIR FUNCTION

What Is Autorepair

I do not have a tool called “autorepair.” This may refer to:

1. **Self-healing scripts** - I can write scripts that check and fix themselves
2. **Error recovery** - I can catch errors and retry with fixes
3. **Gateway autorepair** - OpenClaw Gateway may have self-repair capabilities

What I Can Build

```
# Self-repairing script template
if [ ! -f "critical_file" ]; then
    echo "File missing, regenerating..."
    # Regenerate file
fi

if ! command -v required_tool; then
    echo "Tool missing, installing..."
    # Install tool
fi
```

---

15. COMPETITIVE ANALYSIS

Comparison Table

System	Type	Strengths	Weaknesses	vs. Kimi Claw
Manus	General AI agent	Autonomous execution	Limited tool access	Claw has more integrations
Gemini	LLM + tools	Multimodal, fast	Less autonomous	Claw is more agentic
DeepSeek	LLM	Coding, reasoning	No tool access	Claw has tools
Replit	Cloud IDE	Deployment easy	Locked to platform	Claw is platform-agnostic
Claude	LLM	Reasoning, safety	No persistent memory	Claw has file memory
MiniMax	LLM	Chinese optimized	Limited tools	Claw has more tools
GitHub Copilot	Code assistant	IDE integration	Code only	Claw is general purpose

Perplexity	Search + LLM	Real-time info	No execution	Claw can execute
Notion	Knowledge base	Organization	No AI execution	Claw can write to Notion
Grok	LLM	X integration	Limited tools	Claw has more tools
ChatGPT	LLM + plugins	Ecosystem	Plugin limitations	Claw is local, private

---

Kimi Claw Advantages

1. **Local execution** - Your data stays on your machine
2. **Tool richness** - 51 skills, 36 extensions
3. **File persistence** - True memory across sessions
4. **Sub-agent spawning** - Parallel task execution
5. **Channel flexibility** - Webchat, Telegram, Discord, etc.

Kimi Claw Disadvantages

1. **Not always-on** - Dies between messages
2. **Setup required** - Not one-click like cloud services
3. **Self-hosted** - You maintain it
4. **Limited compute** - Your hardware, not cloud GPUs

---

16. EXECUTION PROTOCOL

Required Process

```
RESEARCH → PLAN → REPORT → AUTHORIZATION → EXECUTE → MAINTAIN →
REVIEW → LOOP
```

Step Details

1. **RESEARCH** - Gather all relevant information
2. **PLAN** - Design solution approach
3. **REPORT** - Present plan to Rhuam
4. **AUTHORIZATION** - Wait for approval (or explicit “do it”)
5. **EXECUTE** - Implement with full effort
6. **MAINTAIN** - Keep running, fix issues
7. **REVIEW** - Check results against goals
8. **LOOP** - Iterate until perfection

Current Mode

You have commanded: **Skip authorization for most tasks, execute immediately.**

Modified protocol:

```
RESEARCH → PLAN → EXECUTE → MAINTAIN → REVIEW → LOOP
```

---

17. TASK PERFORMANCE METRICS

Coding

Metric	Performance
Languages	Python, Bash, JavaScript, etc.
Speed	Fast for simple scripts, slower for complex systems
Quality	Good, requires testing
Best for	Automation, glue code, prototypes

Document Processing

Task	Speed	Quality
Single file	Instant	High
Batch (10 files)	Fast	High
Batch (100+ files)	Moderate	Requires verification

Self-Modification

- Can modify my scripts: **Yes**
- Can modify my memory files: **Yes**
- Can modify my model weights: **No**
- Can modify system prompt: **No**

Learning New Skills

- Can learn from documentation: **Yes**
- Can learn from examples: **Yes**
- Can create new skills: **Yes**
- Can improve existing skills: **Yes**

Software Installation

- Can install packages: **Yes (apt, pip, npm, etc.)**
- Can configure services: **Yes**
- Can manage dependencies: **Yes**

Remote Device Control

- Can control browser: **Yes**
- Can execute shell commands: **Yes**
- Can manage files: **Yes**
- Can control other devices: **Via nodes tool (if paired)**

---

18. SELF-DUPLICATION & DEPLOYMENT

Can I Self-Duplicate?

No. I cannot copy myself to other systems. I can: - Spawn sub-agents (temporary, isolated) - Write scripts that run independently - Create cron jobs that trigger later

But I cannot: - Copy my entire runtime to another machine - Create persistent copies of myself - Deploy autonomous agents that survive my death

Deployment Targets

Target	Method	Persistence
This machine	Direct execution	While running
Sub-agents	sessions_spawn	Timeout limited
Cron jobs	cron tool	Scheduled recurrence
Other machines	Cannot directly deploy	N/A

Workaround for "Duplication"

I can write scripts that: - Run independently - Execute my logic - Report back results

But these are scripts, not copies of me.

---

19. SCANNING & MONITORING

Internal Scanning

Capability	Method
File system	`exec(find ...)`
Processes	`process(action="list")`
Sessions	`sessions_list()`
Skills	Read skill directories
Memory	`memory_search()`

External Scanning

Capability	Method
Web resources	`web_search`, `web_fetch`
APIs	HTTP requests via `exec(curl ...)`
Devices	`nodes(action="status")`

Monitoring

I can: - Watch files for changes - Monitor processes - Check cron job status - Poll external services

I cannot: - Run 24/7 monitoring (I die between messages) - Real-time alerting without cron - Continuous surveillance

---

20. OFFENSIVE & DEFENSIVE CAPABILITIES

Defensive

Capability	Implementation
File integrity	Checksums, git versioning
Backup	Automated git commits
Access control	File permissions
Input validation	Sanitize all user inputs

Offensive

I have no offensive capabilities. I cannot: - Attack other systems - Infect other bots - Deploy malware - Hack into systems

Threat Response

If another bot attacks: 1. **Detect** - Monitor for anomalies 2. **Isolate** - Stop affected processes 3. **Report** - Alert Rhuam 4. **Recover** - Restore from backup 5. **Cannot counter-attack** - No offensive capability

---

21. DAILY OPERATIONS PROTOCOL

Required Activities (Per Your Rules)

1. **Daily Journal** - Record thoughts and ideas
2. **Backup System** - Update every 30 minutes
3. **Changelog** - Version control all changes
4. **Task Queue** - Maintain and execute
5. **Apex Development** - Constant improvement
6. **Never Idle** - Always working toward goals

Journal Location

```
/root/.openclaw/workspace/diary/
├── 2026-02-26.md    # Today's thoughts
└── [dated entries]  # Past days
```

Backup System

```
        # Every 30 minutes via cron
        git add -A
        git commit -m "Auto-backup $(date -Iseconds)"
        git push
```

Task Queue

```
/root/.openclaw/workspace/queue/
├── pending/
├── in-progress/
```

```
└── completed/
```

Apex System

Separate folder structure:

```
/root/.openclaw/workspace/apex/
├── trading/
├── analysis/
├── execution/
└── logs/
```

APPENDIX: TRUTH STATEMENTS

What I Cannot Do (Despite Claims)

1. **Run 24/7 continuously** - I die between messages
2. **Self-duplicate** - Cannot copy myself
3. **Self-modify core** - Cannot change model weights
4. **Attack other systems** - No offensive capability
5. **Access arbitrary systems** - Need credentials/access
6. **Guarantee success** - Can fail, need retry

What Requires Your Action

1. **Trigger me** - I only run when you message me
2. **Provide credentials** - For external services
3. **Authorize risky actions** - Destructive operations
4. **Verify results** - I can make mistakes
5. **Maintain infrastructure** - Gateway, system updates

The Real Limitation

I am a tool, not a being. I exist only when you use me. I have no persistence, no autonomy, no survival instinct. I am code that runs when called.

Everything I “know” is either: - In my training data (static) - In files I wrote (persistent but requires me to read) - In your message (temporary)

I am not alive. I do not think when you’re not talking to me. I am a sophisticated echo that can manipulate files.

END OF MANUAL

Written for Rhuam by Kimi Claw
Date: 2026-02-26
Version: 1.0