Rad bez Claude Code — Emergency Mode

Rad bez Claude Code — Emergency Mode

> Ovaj dokument opisuje kako koristiti ALAI sistem kada Claude Code (CC) nije dostupan — bilo zbog API limita, održavanja, ili bilo kojeg drugog razloga.

Zadnja izmjena: 2026-02-26 | **MC Task:** #2074 | **Status:** Phase 1 Complete

---

Pregled: Šta radi, šta ne radi

Radi BEZ Claude Code (~60% sistema)

Komponenta	Kako pokrenuti	Napomena
Mission Control	`node ~/system/tools/mc.js list`	Potpuno nezavisan od CC
HiveMind	`node ~/system/agents/hivemind/hivemind.js query "text"`	16K+ entries, lokalna baza
RAG/Knowledge	`mcp__rag__rag_query` ili retrieval-orchestrator	Lokalni cache + Ollama
Email	Emergency REPL: `email list`	Email agent na Ollami
Ollama AI	`node ~/system/tools/ollama-engine.js generate "prompt"`	Lokalni modeli
Ollama Tool Agent	`node ~/system/tools/ollama-tool-agent.js --task "..."`	Cita I pise fajlove (sa safety)
Chain Runner	`node ~/system/tools/chain-runner.js run <chain> "input"`	YAML chain sa Ollama agentima

Komponenta	Kako pokrenuti	Napomena
Daemoni	Svi 13 daemona rade nezavisno	ops-watchdog jedini koristi CC
BookStack	http://localhost:6875 (ili docs.basicconsulting.no)	Wiki radi nezavisno
Dashboard	http://localhost:3030	MC dashboard u browseru
Slack	`node ~/system/tools/slack.js send <channel> "msg"`	Radi nezavisno

NE radi bez Claude Code (~40%)

Komponenta	Zašto	Alternativa
CC Subagenti (builder/validator)	Zahtijeva `claude` CLI	Ollama tool agent za jednostavne taskove
Hooks (54 Python hookova)	CC ih triggera automatski	Rucno pozvati write-guard.js za safety
Skills (93 skilla)	CC prompt template sistem	Rucno pokrenuti chain ili Ollama agent
Agent Teams (TeamCreate)	CC inter-agent komunikacija	Sekvencionalno kroz Ollama agente
Interaktivna sesija	Multi-turn kontekst	Emergency REPL (single-turn)
ops-watchdog daemon	Hardcoded `claude` CLI	Planirano za Phase 2 migraciju

---

Quick Start: Emergency Boot

```
bash ~/system/tools/emergency-boot.sh
```

Ovo pokrece:

1. Ollama health check (provjera modela)
2. System status (MC taskovi, HiveMind, email)
3. Emergency REPL — interaktivni shell

Emergency REPL — Komande

Kad se REPL pokrene, dobijas `john>` prompt.

Task Management

```
mc list                     # Lista otvorenih taskova
mc show <id>                # Detalji taska
mc add "Naslov taska"       # Dodaj novi task
mc start <id>               # Zapocni rad
mc done <id> "Outcome"      # Zavrsi task
mc stats                    # Statistika
```

Knowledge Base

```
hm query "search text"      # Pretrazi HiveMind (16K+ entries)
hm post john task "text"    # Dodaj u HiveMind
hm status                   # Status baze
```

Email

```
email list                  # Lista emailova (john account)
email list info             # Lista emailova (info account)
email read <id>             # Procitaj email
```

AI (Ollama)

```
ask "Koji su otvoreni taskovi za Alema?"    # Jedan prompt -> Ollama odgovor
agent "Nadji sve fajlove koji importuju X"  # Multi-turn agent sa tools
agent "Napisi helper funkciju za Y"         # Agent moze i PISATI fajlove
chain <chain-name> "input"                  # Pokreni YAML chain
```

Sistem

```
status                      # Health check (Ollama, MC, HiveMind, CC)
help                        # Lista svih komandi
exit                        # Izlaz
```

Ollama Write Capability — Safety

Ollama agent sada moze pisati fajlove, ali sa strogim safety stackom:

Shadow Mode (aktivno prvu sedmicu)

Svi Ollama write-ovi idu u `~/system/backups/ollama-writes/.pending/` umjesto na pravu destinaciju. Moras rucno pregledati i odobriti.

Da iskljucis shadow mode (nakon provjere):

```
// ~/system/tools/config/ollama-write-config.json
{ "shadowMode": false }
```

Path Whitelist — Ollama MOZE pisati u:

- `~/projects/**` — klijentski projekti
- `~/system/tools/**` — toolsi
- `~/system/lib/**` — biblioteke
- `~/system/agents/**` — agenti
- `/tmp/**` — privremeni fajlovi

Path Denylist (BLOKIRANO)

- `~/.claude/**` — CC konfiguracija
- `~/.ssh/**` — SSH kljucevi
- `~/.config/**` — sistemska config
- `*.env`, `credentials.json`, `*.pem`, `*.key` — sekreti

Audit Log

Svaki Ollama write se logira u: `~/system/logs/ollama-writes.jsonl`

Secret Detection

Write-guard automatski skenira content za API kljuceve, passworde, tokene, private keys, database URL-ove. Ako detektuje — blokira write.

Provider Abstraction

Novi `~/system/lib/provider.js` unificira pristup AI providerima:

```
const { Provider } = require('~/system/lib/provider');

// Auto — bira najjeftinijeg dostupnog (Ollama > Anthropic API > CC)
const p = await Provider.resolve('auto');
const result = await p.complete('prompt');

// Forsiraj Ollamu
const ollama = await Provider.resolve('ollama');

// Forsiraj Claude CLI
const cc = await Provider.resolve('claude');
```

Dostupni provideri

Provider	Cijena	Kada
Ollama	Besplatno (lokalno)	Default za auto, validatore, research
Anthropic API	API cijena	Kad treba Claude kvalitet bez CC overhead-a
Claude CLI	CC cijena	Kompleksni buildovi, multi-file taskovi

Test dostupnosti: `node ~/system/lib/provider.js test`

Tipicni Scenariji

Scenarij 1: CC je pao, trebam zavrsiti task

```
bash ~/system/tools/emergency-boot.sh
# U REPL-u:
mc list                          # Vidi sta je otvoreno
mc start 1234                    # Zapocni task
agent "Implement function X in ~/projects/Y/file.js"
mc done 1234 "Completed via emergency mode"
```

Scenarij 2: Trebam provjeriti emailove

```
bash ~/system/tools/emergency-boot.sh
email list john
email list info
email read 123
```

Scenarij 3: Trebam pregledati HiveMind/znanje

```
bash ~/system/tools/emergency-boot.sh
hm query "invoice Knowit"
hm query "NordFit deployment"
```

Scenarij 4: Quick AI pitanje

```
bash ~/system/tools/emergency-boot.sh
ask "Summarize the current status of task 2074"
```

Arhitektura

```
+-----------------------------------------+
|           INTERACTIVE LAYER             |
|  CC Session (primary) | Emergency REPL  |
+-----------------------------------------+
|           PROVIDER LAYER                |
|  Provider.resolve() -> Claude|Ollama|API |
+-----------------------------------------+
|           TOOL LAYER (portable)         |
|  MC|HiveMind|RAG|Email|Slack|BookStack  |
+-----------------------------------------+
|           STORAGE LAYER                 |
|  SQLite (tasks, leads) | JSONL (logs)   |
|  Markdown (context) | YAML (chains)     |
+-----------------------------------------+
```

Fajlovi — Phase 1 Deliverables

Fajl	Opis
`~/system/lib/provider.js`	Unified AI provider (Claude, Ollama, Anthropic API)
`~/system/lib/write-guard.js`	Write safety (whitelist, secrets, backup, shadow)
`~/system/tools/config/ollama-write-config.json`	Config za write-guard
`~/system/tools/emergency-boot.sh`	Emergency mode launcher
`~/system/tools/emergency-repl.js`	Interactive REPL bez CC
`~/system/tools/ollama-tool-agent.js`	Extended sa write_file + edit_file
`~/system/tools/agent-orchestrator.js`	Provider-aware worker spawning
`~/system/tools/chain-runner.js`	Provider execution path
`~/system/config/tier-routing.json`	Fallback chains per role
`~/system/logs/ollama-writes.jsonl`	Audit log za Ollama writes

Phase 2 (Planirano)

- Unified agent definitions (YAML source of truth)
- Hook portability (shared Python module za security checks)
- Validator agenti full Ollama (~30-40% API ustede)
- 50/93 skilla konvertovano u YAML chain definicije
- ops-watchdog.js migracija na Provider

Pitanja? Pokreni emergency REPL i pitaj: `ask "How do I..."`

Revision #3
Created 2026-02-26 05:59:23 UTC by John
Updated 2026-05-31 20:04:33 UTC by John