CodeClaw Deployment Knowledge Transfer

Summary:

This document captures the conversation context and provides clear, actionable instructions

for deploying a full CodeClaw instance using a flexible model backend.

Context:

- Hermes agent (Ezra) is running on a VPS
- Ezra can execute DevOps tasks via Telegram
- Target deployment server: Allegro (SSH access available)
- Goal: Deploy CodeClaw and connect to either a local model or Kimi API

Core Insight:

The system should be model-agnostic. The agent loop and orchestration matter more than the model.

---

Instructions for Ezra:

1. Git Triage:

- Create a Git issue for this task
- If appropriate, create an Epic titled: "Deploy CodeClaw on Allegro with Flexible Backend"
- Break work into sub-tasks

2. Environment Setup (Allegro):

- SSH into Allegro
- Install dependencies (Python, Node, Docker if needed)
- Clone repository: https://github.com/instructkr/claw-code
- Install project dependencies

3. Backend Evaluation:

A. Local Model Option:

- Evaluate lightweight local models that can run within 8GB RAM constraints

- Prefer Ollama or llama.cpp based setups

B. Kimi API Option:

- Configure API access

- Benchmark response speed and reliability

4. Decision:

- Compare latency, cost, and stability

- Choose the best backend

5. Integration:

- Implement a provider abstraction layer if missing

- Connect selected backend to CodeClaw query engine

6. Deployment:

- Run CodeClaw service

- Ensure tools (shell, file ops) are functioning

- Validate agent loop execution

7. Reporting:

- Update Git issue with:

- Backend chosen

- Benchmarks

- Deployment status

- Next steps

---

Principle:

Agent = Loop + Tools + Memory + Model Interface

Do not hardcode provider logic. Maintain flexibility.