PHASE 4 — SYSTEM POWER & RESILIENCE

“Antigravity can die. CLAWBOLT does not.”

This phase gives you:

- Controlled restarts
- Health reporting
- File system access
- Polling + auto-rebind logic

No AI dependency. No UI dependency.

🎯 PHASE 4 OBJECTIVES

✅ /restart — restart Antigravity only

✅ Poll Antigravity until alive

✅ Auto-rebind input mapping after restart

✅ /report — full system health

✅ /ls <path> — remote filesystem view

✅ Safe confirmations for destructive actions (prep for Phase 5)

📁 DIRECTORY EXPANSION (PHASE 4)

```
agent/
├── restart.py
├── report.py
├── ls.py
```

1️⃣

/restart

— ANTIGRAVITY CONTROL (CRITICAL)

Behavior

- Close Antigravity
- CLAWBOLT keeps running
- Poll every N seconds
- When Antigravity appears:
    - Focus window
    - Re-map input box
    - Notify Telegram

agent/restart.py

```
import subprocess
import time
from antigravity.window import focus_antigravity

ANTIGRAVITY_CMD = ["pkill", "-f", "Antigravity"]

async def restart_antigravity(update):
    await update.message.reply_text("🔄 Restarting Antigravity...")

    subprocess.run(ANTIGRAVITY_CMD)

    await update.message.reply_text("⏳ Waiting for Antigravity to
reopen...")

    for _ in range(30):  # ~30 seconds
        time.sleep(1)
        if focus_antigravity():

            await update.message.reply_text("✅ Antigravity
detected and reattached")
            return

    await update.message.reply_text("⚠️ Antigravity did not reopen
in time")
```

2️⃣

/report

— SYSTEM HEALTH SNAPSHOT

This is non-negotiable for remote ops.

What it reports

- OS
- CPU / RAM usage
- Disk usage
- Uptime

- Antigravity status

agent/report.py

```
import platform
import psutil
import subprocess

async def system_report(update):
    cpu = psutil.cpu_percent()
    mem = psutil.virtual_memory().percent
    disk = psutil.disk_usage("/").percent

    ag_status = subprocess.run(
        ["pgrep", "-f", "Antigravity"],
        stdout=subprocess.DEVNULL
    )
    ag_running = "RUNNING" if ag_status.returncode == 0 else
"STOPPED"

    report = f"""
🖥️ SYSTEM REPORT

OS: {platform.platform()}
CPU Usage: {cpu}%
RAM Usage: {mem}%
Disk Usage: {disk}%
Antigravity: {ag_running}
"""
    await update.message.reply_text(report)
```

3️⃣

/ls <path>

— REMOTE FILE VIEW

This is dangerous but powerful, so read-only.

agent/ls.py

```
import os

async def list_dir(update, text):
    parts = text.split(maxsplit=1)
    path = parts[1] if len(parts) > 1 else "."

    if not os.path.exists(path):
        await update.message.reply_text("❌ Path not found")
        return

    if os.path.isfile(path):
        await update.message.reply_text("❌ Path is a file")
        return

    files = os.listdir(path)
    output = "\n".join(files[:50])

    await update.message.reply_text(f"📂 {path}\n\n{output}")
```

4️⃣ UPDATE COMMAND ROUTER

agent/router.py

(UPDATED)

```
from agent.rules import show_rules
from agent.screen import do_screen
from agent.restart import restart_antigravity
from agent.report import system_report
from agent.ls import list_dir

COMMANDS = {
    "/rules": show_rules,
    "/screen": do_screen,
    "/restart": restart_antigravity,
    "/report": system_report,
    "/ls": list_dir,
}
```

```
async def route_command(update, text):
    cmd = text.split()[0]
    handler = COMMANDS.get(cmd)

    if not handler:

        await update.message.reply_text("❌ Unknown command. Use
/rules")
        return

    await handler(update, text)
```

🧪 FAILURE SCENARIOS (INTENTIONAL TESTS)

Scenario	Expected Result
Antigravity frozen	/restart recovers
Antigravity closed	/screen still works
High CPU	/report shows it
Wrong path /ls	Safe error
Antigravity UI moved	Rebind on next AI send

If these pass → CLAWBOLT is resilient.

🧠 WHAT YOU HAVE NOW

At this point, CLAWBOLT can:

✅ Control AI

✅ Ignore AI

✅ Survive crashes

✅ Report system health

✅ Manipulate the environment

This is already beyond most “AI agents”.