GNU Group — Linux Production Incident Investigation

Learner field workbook · Pilot module 01 · v1.0

Incident ID: LPI-01 · **Scenario:** The healthy service with failing orders · **Difficulty:** Intermediate · **Time:** 180 minutes instructor-led

Operational safety: Run only on a disposable training VM or your own workstation, as an ordinary Linux user. Do not run with sudo. The lab binds to 127.0.0.1; it does not create systemd units, alter kernel limits globally or touch outside services. All names and orders are fictional.

Outcome and evidence standard

You will: (1) establish the customer-visible symptom, (2) separate observation from hypothesis, (3) find the process/resource constraint using real Linux evidence, (4) explain why a shallow health check missed it, (5) recommend mitigation and durable repair, and (6) demonstrate a before/after verification trail. Every conclusion needs a specific command, log line, or observed HTTP result.

Task 0 — Entry skills (10 min)

Confirm you can run python3, use ps, read /proc/PID, and distinguish HTTP 200 from HTTP 503. Write one sentence explaining why an HTTP 200 health check may miss application failures.

Task 1 — Establish the baseline (15 min)

Run:

```
python3 labctl.py start --mode broken
python3 labctl.py status
curl -sS -i http://127.0.0.1:18081/health
curl -sS -i http://127.0.0.1:18081/api/orders
```

Record: UTC timestamp; service PID; current health status; business endpoint result. If curl is absent, use python3 -m urllib.request only after discussing the alternative with the instructor; the harness can probe using python3 labctl.py probe without curl.

Task 2 — Reproduce under controlled load (15 min)

```
python3 labctl.py probe --label broken --requests 50
curl -sS -i http://127.0.0.1:18081/health
curl -sS -i http://127.0.0.1:18081/api/orders
```

Record: counts of HTTP 200/503, business impact, whether /health still reports healthy. Do not claim the issue is network-wide: the simulator uses local loopback only.

Task 3 — Collect Linux process evidence (20 min)

```
python3 labctl.py status
# Substitute the PID printed by status:
ps -p <PID> -o pid,ppid,stat,etime,cmd
cat /proc/<PID>/limits | grep -i 'open files'
find /proc/<PID>/fd -mindepth 1 -maxdepth 1 | wc -l
cat .runtime/service.log
python3 labctl.py capture --label broken
```

Record: process limit, FD count, error number and error text. Why does errno 24 matter? Which observation proves the failure is not simply a DNS problem, firewall block or bad port?