Project 01 — Red/Blue Oncall Game: Incident-in-a-Box × Observability Stack

This is a **Red/Blue** capstone project. Teams are **1–2 students** each.

- **Red team (scenario authors):** build *diagnosable* incidents + scoring.
- **Blue team (responders):** build an observability/diagnosis workflow that can solve unknown incidents.

All teams must work within a **laptop + Ubuntu VM** budget and produce a reproducible artifact plus a report (60%) and oral defense/demo (40%).

> ***Baseline expectation (this course): MS level.***

Learning goals

Connect **measurement → mechanism** under realistic constraints:

- choose VM-friendly signals (PSI, /proc, cgroup v2, logs, tracepoints/eBPF where possible)
- build an evidence chain (not guesses)
- practice “systems narrative”: timeline, hypothesis, falsification, mitigation

---

Pairing and exchange formats

When you pick this project, we will try to assign blue/red teams for you. If you have a strong preference, please note that in the Canvas submission.

What you must deliver (MS baseline; both teams)

- A runnable repo that can reproduce the incident(s) in a clean VM.
- A **reproduction script** (one command) that:
  1. starts the target service/workload,
  2. starts metrics/log collection,
  3. triggers the incident,
  4. outputs an artifact bundle (raw logs + metrics + plots).
- Technical report (**60%**) + oral report/demo (**40%**).

For every scenario (red-authored, blue-solved), the final writeup must include:

- **2 independent supporting signals** (signals from different observation layers) *plus* **1 negative control / exclusion** (alternative but wrongful explainations)
- **before/after p50/p95/p99** (or p50/p99 at minimum) *plus* **one mechanism-level metric that moved**

Required scenario coverage (3 buckets)

You will implement **3 scenarios**. They must collectively cover:

1. **Scheduling / concurrency** (Week 3–4)
   - examples: runqueue delay, noisy neighbor on same CPU, lock convoying

2. **cgroup v2 resource boundary** (Week 5–6)
    - examples: CPU quota throttling (`cpu.max`), memory limits (`memory.max`) → reclaim/OOM
3. **Storage / writeback tail latency** (Week 8)
    - examples: background `fsync` writer, writeback bursts, IO saturation

You still must follow the difficulty ladder:

1. single root cause, obvious signals
2. two interacting causes
3. misleading primary symptom (root cause elsewhere)

Red team deliverables (scenario pack)

Each scenario must include:

- **Symptom spec:** what alert(s) fire; what the user sees (e.g., p99 latency spike, error rate).
- **Ground truth:** a precise causal story tied to OS/runtime mechanisms.
- **Injection mechanism:** scripts/configs to reliably trigger it in the VM.
- **Evidence contract (MS baseline):**
    - **2 independent supporting signals** (VM-friendly)
    - **1 negative control / exclusion** (explicitly rule out a plausible alternative)
- **Anti-guessing guardrails:** solvable using the evidence contract; avoid “magic outputs”.

Scoring interface

Provide a simple scoring method (can be crude):

- checklist-style points for collecting key evidence
- points for correct root cause + correct mitigation
- penalties for “unnecessary disruptive actions” (e.g., restarting everything without evidence)

Blue team deliverables (diagnosis workbench)

Build a diagnosis workflow that can solve incidents you did not author:

- a minimal “workbench” that presents:
    - how to run the workload
    - how to collect signals
    - how to pivot from symptom → suspect resource → mechanism
- a dashboard/report generator (one of):
    - notebook + saved figures
    - a lightweight web dashboard
    - Prometheus/Grafana (only if you can keep it small)

Required case outcomes (MS baseline)

For each of the 3 red-team scenarios:

- a timeline of observed signals
- a root cause statement (mechanism-level)
- at least **one mitigation** and validation with:
    - before/after p50/p95/p99

- a mechanism metric moving in the expected direction
- one explicit negative control / exclusion

Allowed signals/tools (VM-friendly default set)

- Latency metrics: wrk/hey/vegeta outputs, app histograms, logs
- Host/container: `pidstat`, `vmstat`, `iostat`, `sar`, `ss`, `top -H`
- Kernel interfaces: `/proc/pressure/*` (PSI), `/proc/vmstat`, `/proc/schedstat` (if available)
- cgroup v2: `cpu.stat`, `memory.current`, `memory.stat`, `io.stat`, `cgroup.events`
- Optional: bpftrace/BCC on tracepoints (only if permitted in your VM)

Constraint reminder: **perf hardware PMU counters may be unavailable in VMs**. Do not make your core evidence chain depend on PMU cache counters.

Scaffolding (optional, recommended)

We provide starter templates:

- Red team scaffold: `red_team/`
- Blue team scaffold: `blue_team/`

They include scenario templates and VM-friendly collectors/parsers you can adapt.

Grading

See **`RUBRIC.md`** for the detailed, checkable rubric and “what counts as core difficulty” for this project.