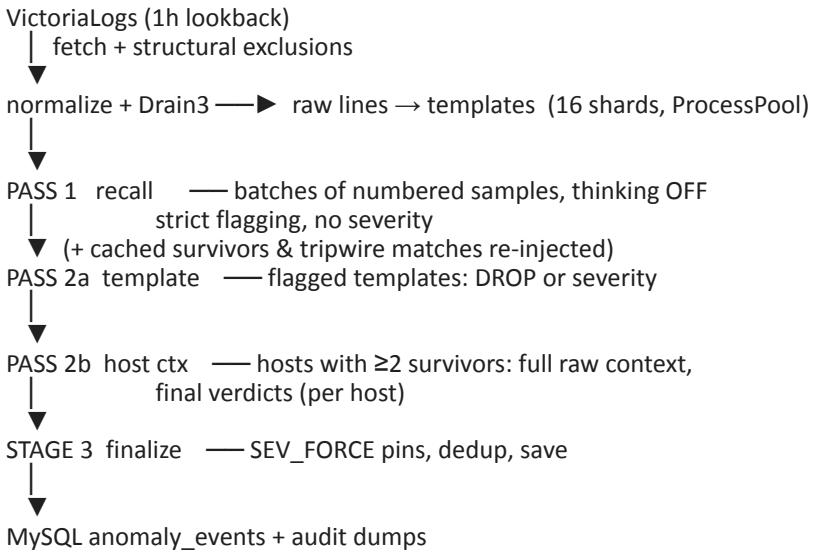
Flow v5.5

An autonomous **hourly log-triage loop** for SDCC (~2500 hosts, containers, openshift):

- Pulls the last hour of logs from **VictoriaLogs**
- Groups millions of raw lines into **templates** (Drain3)
- Sends candidates to a **served LLM** (`nemotron-3-ultra-550b-nvfp4` via the shared inference endpoint) in a 3-stage funnel
- Saves genuine incidents to **MySQL** (`anomaly_events`) → Grafana dashboards



v5.5 PASSES

PASS 1:

- is a fast first sweep of everything in the last hour
- it flags anything that might be a real failure, bias to overflag rather than miss
- it's cheap because it only outputs line numbers

PASS 2a:

- is a strict noise filter
- it judges each template from from PASS 1 and decides:
  - drop as routine or assign severity
- it judges with more context that pass1:
  - each template arrives as [N hosts, appname=app]
  - how many hosts saw it and which application produce it are strong signals (one-host error vs same line on 300 hosts)

PASS 2b:

- is the context judge
- it takes the survivors on each host and re-reads them together as one host story
- IDEALLY:
  - a line that looked alarming alone may be dropped as harmless side-effect of the real problem next to it
  - the same template can be CRITICAL on one host but noise on the other
- it only runs for nodes that have at least 2 lines to judge (no context on 1 line)

v5.5 Safe guards

Why we don’t rely on the AI alone:

- The AI is good at judging new, ambiguous messages - but it's probabilistic: it can miss things, change its mind, or regress after an update.
- A system that alerts humans needs guarantees, not probabilities. So the known stuff is handled deterministically, and the AI only earns its keep on the gray areas.

Pins:

- Some failures are unambiguous: "out of memory", "disk full", "core dumped". Asking an AI to re-derive what a regex already knows wastes money and adds risk.
- What they do: if a line matches a known failure pattern, its severity is hard-wired — no AI call, and the AI can never downgrade or silence it.
- ex: (“No space left”, “CRITICAL”)

Tripwire:

- if pass1 misses some known rules (tripwire) it picks them up and sends them to 2a
- ex: (“LustreError”, None)

Verdict cache - memory

- Every unique message gets judged once; the verdict is remembered.
- Three payoffs:
  - Cost — routine noise is never re-judged, cycle after cycle
  - Reliability — known problems are saved every hour even if the AI misses them
  - Consistency — the same line gets the same answer every time; no judgment lottery

Examples - the one-slide takeaway

```
spine:  fetch → template → PASS 1 → [rescue / cache] → 2a → 2b → SAVE
─────────────────────────────────────────────────────────────────────

1. PIN — "EXT4-fs (dm-0): No space left on device"  (disk_full)

   fetch   ──► template: "EXT4-fs (dm-<*>): No space left on device"
   pass 1  ──► flagged ✓   (if missed, the tripwire rescues it anyway)
   2a      ──► tripwire match → CRITICAL pinned, no AI call
   2b      ──► says "MEDIUM" — fs recovered, per-host opinion
   SAVE    ──► pin overrides: saved as CRITICAL. The AI never has the last word.

─────────────────────────────────────────────────────────────────────

2. TRIPWIRE — "LustreError: 11421:0:(mdc_locks.c): ..."  (tripwire-only, no pin)

   fetch   ──► template: "LustreError: ..."
   pass 1  ──► ✗ MISSED
   rescue  ──► tripwire regex matches → injected back into the funnel
               (stats: tripwire_rescued: 1)
   2a      ──► deterministic HIGH (tripwire fallback), no AI call
   2b      ──► judges per host as usual
   SAVE    ──► saved — and strict_hits["LustreError"] += 1  (the heartbeat)

─────────────────────────────────────────────────────────────────────

3. CACHE — "systemd: Started Session 4213 of user root."  (recurring noise)

   cycle 1:  pass 1 flags → 2a (AI) says DROP → written to cache:
             t:"started session n of user root" = DROP
   cycle 2+: pass 1 flags it again? → 2a reuses the cached DROP, zero AI cost
             pass 1 misses it?      → never enters the funnel at all, zero work
   daily:    whole table expires (TTL 24h) → one cold re-learn cycle, then stable

─────────────────────────────────────────────────────────────────────

4. DROP — "CRON[8811]: (root) CMD (/usr/bin/updatedb)"  (brand-new noise)

   fetch   ──► template: never seen before
   pass 1  ──► flagged ✓ (over-flagging is pass 1's job)
   2a      ──► AI judges: DROP → audit trail: {"stage": "2a", "verdict": "DROP"}
   SAVE    ──► nothing saved this cycle — but the verdict lands in the cache,
               so from now on this line lives journey #3

─────────────────────────────────────────────────────────────────────

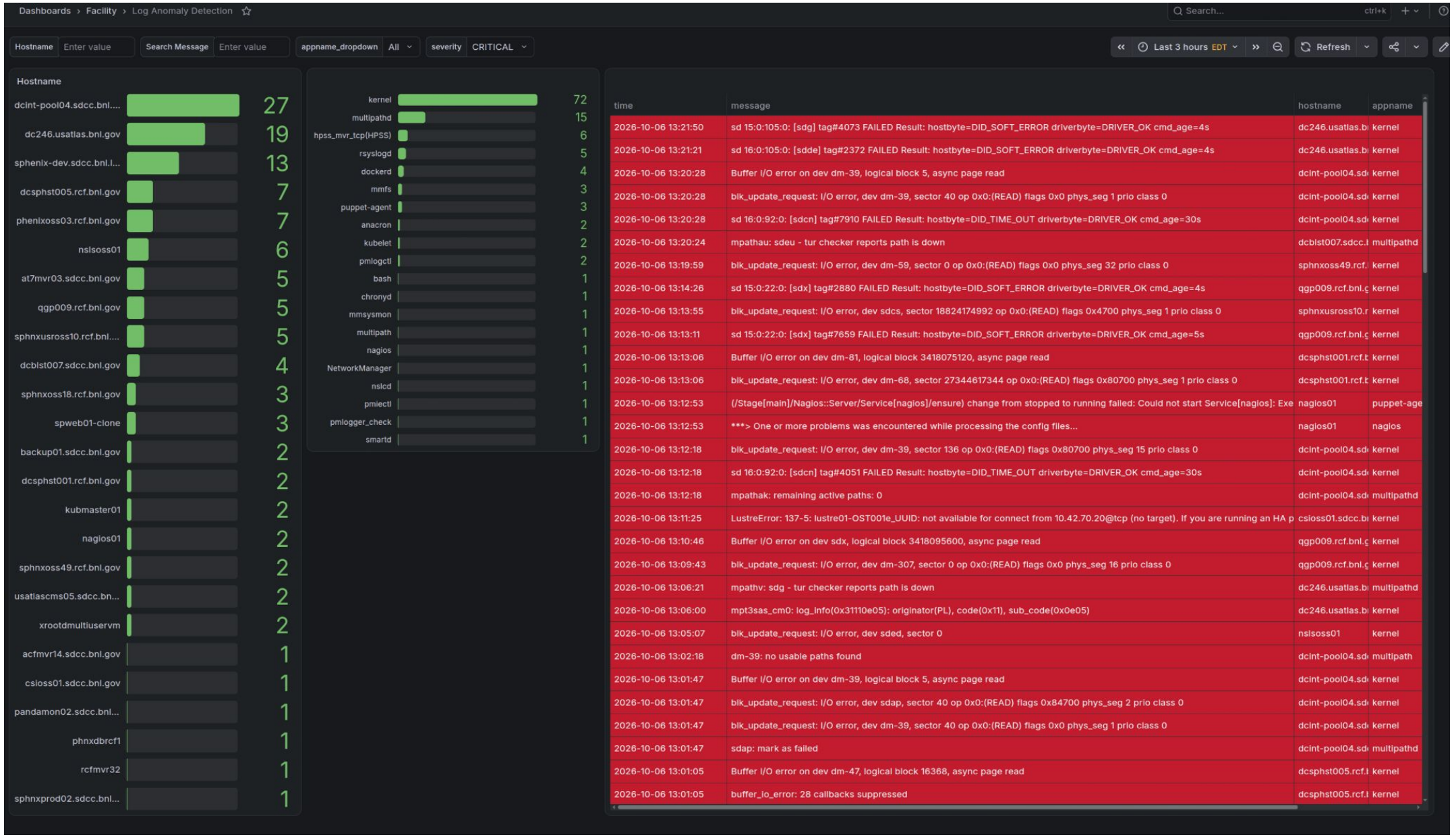
5. NORMAL IDENTIFIED PROBLEM — "nfs: server nfs01 not responding, still trying"
   (no pin, no tripwire, never seen before — the AI's bread and butter)

   fetch   ──► template: "nfs: server nfs-<*> not responding, still trying"
   pass 1  ──► flagged ✓ — strict contract catches it: real storage failure
   2a      ──► fresh template → AI judges with context: [17 hosts, appname=kernel]
               "host can't reach storage" → CRITICAL (verdict cached)
   2b      ──► (only if this host has ≥2 survivors) reads the host's full story —
               e.g. "failing over to backup path" nearby → may adjust per host
   SAVE    ──► saved at the per-host verdict

─────────────────────────────────────────────────────────────────────

THE LOOP:  today's journey #5 is tomorrow's journey #3 — judged once, remembered.
NOTE (v55): pins protect severity, not survival — only mce_edac survives a 2b
DROP (PIN_BEATS_DROP); a 2a DROP kills any template outright.
```

GRAFANA

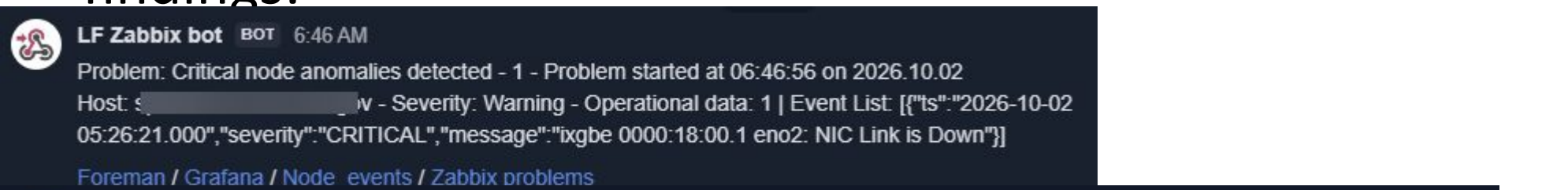

Dashboards › Facility › Log Anomaly Detection
Search...
ctrl+k
Hostname
Enter value
Search Message
Enter value
appname_dropdown
All
severity
CRITICAL
Last 3 hours EDT
Refresh
Hostname
dcint-pool04.sdcc.bnl.... 27
dc246.usatlas.bnl.gov 19
sphenix-dev.sdcc.bnl.l... 13
dcsphst005.rcf.bnl.gov 7
phenixoss03.rcf.bnl.gov 7
nslsoss01 6
at7mvr03.sdcc.bnl.gov 5
qgp009.rcf.bnl.gov 5
sphnxusross10.rcf.bnl.... 5
dcblst007.sdcc.bnl.gov 4
sphnxoss18.rcf.bnl.gov 3
spweb01-clone 3
backup01.sdcc.bnl.gov 2
dcsphst001.rcf.bnl.gov 2
kubmaster01 2
nagios01 2
sphnxoss49.rcf.bnl.gov 2
usatlascms05.sdcc.bn... 2
xrootdmultiuservm 2
acfmvr14.sdcc.bnl.gov 1
csloss01.sdcc.bnl.gov 1
pandamon02.sdcc.bnl... 1
phnxdbrcf1 1
rcfmvr32 1
sphnxprod02.sdcc.bnl... 1
kernel 72
multipathd 15
hpss_mvr_tcp(HPSS) 6
rsyslogd 5
dockerd 4
mmfs 3
puppet-agent 3
anacron 2
kubelet 2
pmlogctl 2
bash 1
chronyd 1
mmsysmon 1
multipath 1
nagios 1
NetworkManager 1
nslcd 1
pmiectl 1
pmlogger_check 1
smartd 1
time
message
hostname
appname
2026-10-06 13:21:50 sd 15:0:105:0: [sdg] tag#4073 FAILED Result: hostbyte=DID_SOFT_ERROR driverbyte=DRIVER_OK cmd_age=4s dc246.usatlas.b kernel
2026-10-06 13:21:21 sd 16:0:105:0: [sdde] tag#2372 FAILED Result: hostbyte=DID_SOFT_ERROR driverbyte=DRIVER_OK cmd_age=4s dc246.usatlas.b kernel
2026-10-06 13:20:28 Buffer I/O error on dev dm-39, logical block 5, async page read dcint-pool04.sd kernel
2026-10-06 13:20:28 blk_update_request: I/O error, dev dm-39, sector 40 op 0x0:(READ) flags 0x0 phys_seg 1 prio class 0 dcint-pool04.sd kernel
2026-10-06 13:20:28 sd 16:0:92:0: [sdcn] tag#7910 FAILED Result: hostbyte=DID_TIME_OUT driverbyte=DRIVER_OK cmd_age=30s dcint-pool04.sd kernel
2026-10-06 13:20:24 mpathau: sdeu - tur checker reports path is down dcblst007.sdcc.l multipathd
2026-10-06 13:19:59 blk_update_request: I/O error, dev dm-59, sector 0 op 0x0:(READ) flags 0x0 phys_seg 32 prio class 0 sphnxoss49.rcf. kernel
2026-10-06 13:14:26 sd 15:0:22:0: [sdx] tag#2880 FAILED Result: hostbyte=DID_SOFT_ERROR driverbyte=DRIVER_OK cmd_age=4s qgp009.rcf.bnl.g kernel
2026-10-06 13:13:55 blk_update_request: I/O error, dev sdcs, sector 18824174992 op 0x0:(READ) flags 0x4700 phys_seg 1 prio class 0 sphnxusross10.r kernel
2026-10-06 13:13:11 sd 15:0:22:0: [sdx] tag#7659 FAILED Result: hostbyte=DID_SOFT_ERROR driverbyte=DRIVER_OK cmd_age=5s qgp009.rcf.bnl.g kernel
2026-10-06 13:13:06 Buffer I/O error on dev dm-81, logical block 3418075120, async page read dcsphst001.rcf.b kernel
2026-10-06 13:13:06 blk_update_request: I/O error, dev dm-68, sector 27344617344 op 0x0:(READ) flags 0x80700 phys_seg 1 prio class 0 dcsphst001.rcf.b kernel
2026-10-06 13:12:53 (/Stage[main]/Nagios::Server/Service[nagios]/ensure) change from stopped to running failed: Could not start Service[nagios]: Exe nagios01 puppet-age
2026-10-06 13:12:53 ***> One or more problems was encountered while processing the config files... nagios01 nagios
2026-10-06 13:12:18 blk_update_request: I/O error, dev dm-39, sector 136 op 0x0:(READ) flags 0x80700 phys_seg 15 prio class 0 dcint-pool04.sd kernel
2026-10-06 13:12:18 sd 16:0:92:0: [sdcn] tag#4051 FAILED Result: hostbyte=DID_TIME_OUT driverbyte=DRIVER_OK cmd_age=30s dcint-pool04.sd kernel
2026-10-06 13:12:18 mpathak: remaining active paths: 0 dcint-pool04.sd multipathd
2026-10-06 13:11:25 LustreError: 137-5: lustre01-OST001e_UUID: not available for connect from 10.42.70.20@tcp (no target). If you are running an HA p csloss01.sdcc.b kernel
2026-10-06 13:10:46 Buffer I/O error on dev sdx, logical block 3418095600, async page read qgp009.rcf.bnl.g kernel
2026-10-06 13:09:43 blk_update_request: I/O error, dev dm-307, sector 0 op 0x0:(READ) flags 0x0 phys_seg 16 prio class 0 qgp009.rcf.bnl.g kernel
2026-10-06 13:06:21 mpathv: sdg - tur checker reports path is down dc246.usatlas.b multipathd
2026-10-06 13:06:00 mpt3sas_cm0: log_info(0x31110e05): originator(PL), code(0x11), sub_code(0x0e05) dc246.usatlas.b kernel
2026-10-06 13:05:07 blk_update_request: I/O error, dev sded, sector 0 nslsoss01 kernel
2026-10-06 13:02:18 dm-39: no usable paths found dcint-pool04.sd multipath
2026-10-06 13:01:47 Buffer I/O error on dev dm-39, logical block 5, async page read dcint-pool04.sd kernel
2026-10-06 13:01:47 blk_update_request: I/O error, dev sdap, sector 40 op 0x0:(READ) flags 0x84700 phys_seg 2 prio class 0 dcint-pool04.sd kernel
2026-10-06 13:01:47 blk_update_request: I/O error, dev dm-39, sector 40 op 0x0:(READ) flags 0x0 phys_seg 1 prio class 0 dcint-pool04.sd kernel
2026-10-06 13:01:47 sdap: mark as failed dcint-pool04.sd multipathd
2026-10-06 13:01:05 Buffer I/O error on dev dm-47, logical block 16368, async page read dcsphst005.rcf.l kernel
2026-10-06 13:01:05 buffer_io_error: 28 callbacks suppressed dcsphst005.rcf.l kernel

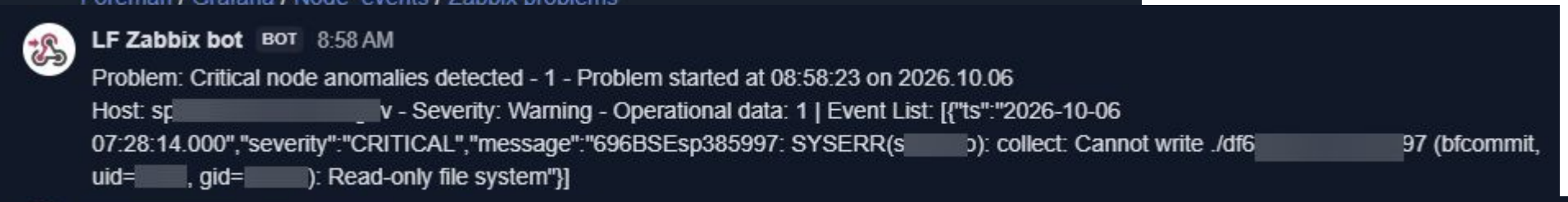
AI findings are not sent as reports. Separate script is scraping the DB on the Zabbix server. It stores all the findings, but only CRITICAL findings are enabled to send alerts

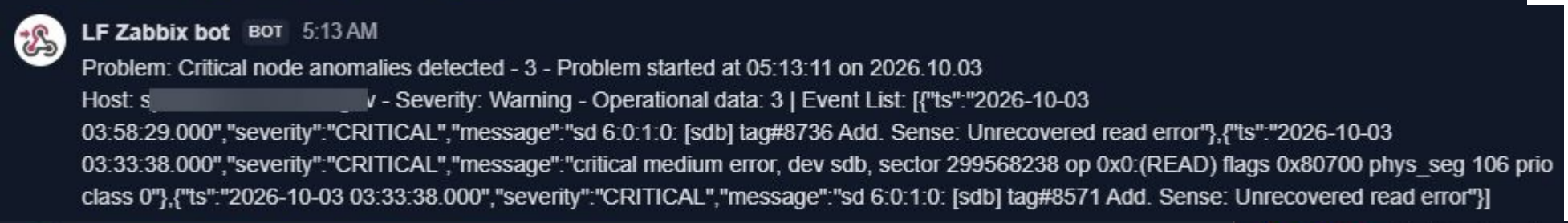
Name ▲	Triggers	Key
AI Anomalies		check_ai_anomaly.py["-H","{HOST.HOST}","-t","2"]
AI Anomalies: AI Anomalies - anomaly list		ai-anomaly-list
AI Anomalies: AI Anomalies - CRITICAL count	Triggers 1	ai-critical-count
AI Anomalies: AI Anomalies - critical list	Triggers 1	ai-critical-list
AI Anomalies: AI Anomalies - HIGH count	Triggers 1	ai-high-count
AI Anomalies: AI Anomalies - high list		ai-high-list
AI Anomalies: AI Anomalies - LOW count	Triggers 1	ai-low-count
AI Anomalies: AI Anomalies - low list		ai-low-list
AI Anomalies: AI Anomalies - MEDIUM count	Triggers 1	
AI Anomalies: AI Anomalies - medium list		
AI Anomalies: AI Anomalies - OOM count	Triggers 1	
AI Anomalies: AI Anomalies - oom list		

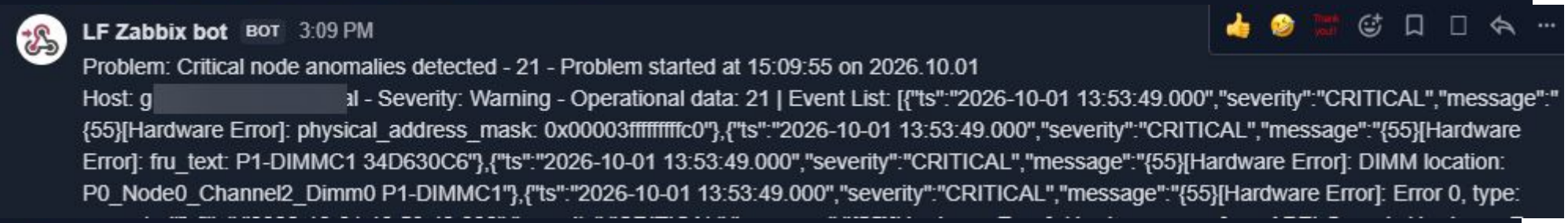
```
[root@l        1 ~]#                                  check_ai_anomaly.py -H s                                  v -t 6|jq
{
  "num_of_node_anomalies": {
    "CRITICAL": 3,
    "HIGH": 16,
    "MEDIUM": 18,
    "OOM": 4
  },
  "anomalies": {
    "MEDIUM": [
      {
        "ts": "2026-10-06 13:05:08.000",
        "severity": "MEDIUM",
        "message": "osc: Process '/usr/sbin/lctl set_param 'osc.sp      -OST026c-osc-MDT0000.active=1'' failed with exit code 2."
      },
      {
        "ts": "2026-10-06 11:35:05.000",
        "severity": "MEDIUM",
        "message": "(sphenix.sdcc.bnl.gov) looks like cvmfs has been crashed previously"
      },
      {
        "ts": "2026-10-06 11:35:03.000",
        "severity": "MEDIUM",
        "message": "[E] Killing connection from 1          7, err 120"
      },
      {
        "ts": "2026-10-06 11:20:31.000",
        "severity": "MEDIUM",
        "message": "[A] Could not initialize handlers for compression libraries: z lz4 zfast alphae alphah"
      },
```

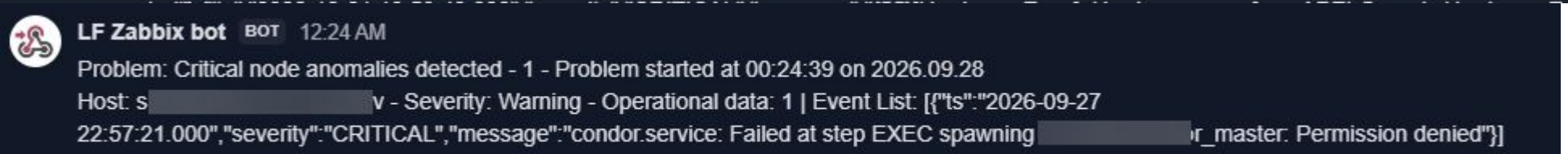
Critical alerts are being sent to Mattermost channel.Some interesting findings:











Flow v5.9

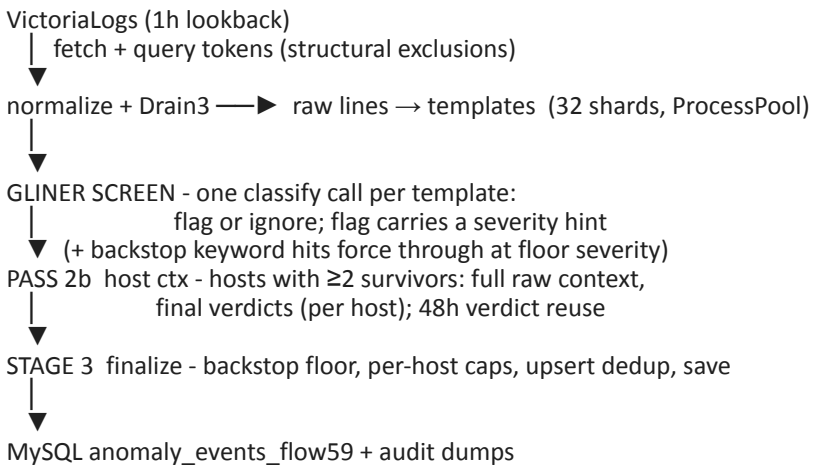
An autonomous **hourly log-triage loop** for SDCC (~2500 hosts, containers, openshift):

Pulls the last hour of logs from **VictoriaLogs**

Groups millions of raw lines into **templates** (Drain3)

Sends candidates to 2 **served LLM** (GLiNER2.5 and NVIDIA-Nemotron-3-Nano-4B) in a 2-stage funnel

Saves genuine incidents to **MySQL** (`anomaly_events`) → Grafana dashboards



v5.9 PASSES

GLiNER SCREEN (one model — replaces PASS 1 + PASS 2a):

- one cheap call per first-seen template: the raw sample line alone decides flag + severity hint; ignore/DROP templates die here
- still biased to overflag rather than miss; no host/appname context at this stage — that context moved into 2b
- backstop: known-failure keywords force the template through at a floor severity even if the screener said ignore or DROP

PASS 2b (unchanged contract):

- the context judge: survivors on each host re-read together as one host story
- each template arrives as N occurrences, appname — one-host error vs the same line on 300 hosts
- IDEALLY:
  - a line that looked alarming alone may be dropped as a harmless side-effect of the real problem next to it; the same template can be CRITICAL on one host and noise on another
  - only runs for hosts with ≥2 lines to judge (no context on 1 line)

STAGE 3 (finalize):

- 2b may escalate context but never DROP a backstop family — severity is floored
- per-host entry caps: one fleet-wide incident can't starve other hosts' saves
- one row per (host, template), latest occurrence refreshes ts/severity — the DB row is the dedup

v5.9 Safe guards/improvements

The policy lives in the models - trained, not programmed:

flow55 wrapped general-purpose models with severity pins, tripwires, cache rules and a growing rule file - code compensating for what the models didn't know. flow59 replaces that with two finetuned models, trained on our own labeled data (36,003 log families from our infrastructure, audited into a ground truth):

- GLiNER2.5-Decide → g_v10 (screen): learned which templates are signal and what severity they carry — one call per template, no rule plumbing around it.
- NVIDIA-Nemotron-3-Nano-4B-BF16 → n_v20 (judge): learned our severity rubric with host context — per-host verdicts from the model, not prompt tricks or post-hoc patching.

And 550B → 4B + 0.5B on one GPU:

- the 550B model pipeline ran on at least 8×H100
- head-to-head on the same log sample: it missed roughly half the lines of interest - recall 48–56% vs 98% for the finetuned screener
- what it did catch it frequently mislabeled - 2-3x worse false-positive rate
- the two finetuned successors host screen + judge together on a single H100 — better coverage, learned from our site own data, at a fraction of the hardware

What stays deterministic is deliberately minimal:

- backstop - a 7-family regex floor (oom-killer, disk full, hardware error…) so the known silent-failure classes can never be silenced by any model
- query tokens - a handful of known-noise tokens at fetch

A new failure pattern enters by becoming a training row — not a new rule.

Hardware — what it took to get to v5.9:

6× H100 — the workbench (training + serving for every measurement; prod never runs here)

- nano4b (4B) finetune: ~2h per run across all 6 GPUs (TP=6, 732 steps)
- GLiNER finetune: ~30 min on a single GPU (486M params)
- between runs the same box hosts both models for the val benchmarks, the 10-window backtests, and smokes (judge TP=4, screener 1 GPU)

4× RTX PRO6000 — the engineer:

- hosts GLM-5.3-Flash, the opencode agent model that did the engineering and that wrote everything around the models
- engine sweep
- eval harness
- ground-truth consolidation
- benchmark gates
- live comparisons
- putting up with me

no GPU needed for: dataset builds, ground-truth consolidation, scoring, the DB, and the replay drivers (CPU, dev box)