SYNTHETIC SAMPLE. Generated from hand-authored lab fixtures. No customer, no real system, no measured outcomes.

VitaOps Intelligence

Java/Linux Production Reliability Assessment — Sample Report

Parser 0.2.0 · ruleset 0.2.0 · Environment: synthetic lab

1. Executive summary

This sample shows the structure of a fixed-scope assessment. In a real engagement the scope (one environment, agreed nodes, data window) is set in a signed statement of work and data agreement. Here, 11 synthetic scenarios were analyzed offline by deterministic rules. Confidence labels are rule categories, not probabilities. Business impact is **not measured** in this sample.

Scenario overview

Scenario	Status	Top hypothesis	Confidence
Intermittent slowness with errors from the orders datasource	Supported	JVM heap pressure; memory leak not confirmed	Moderate
Order service latency spikes during business hours	Supported	JVM heap pressure; memory leak not confirmed	Moderate
Morning import job: long pauses that stop after lunch	Supported	JVM heap pressure; memory leak not confirmed	Low
Checkout requests time out under load	Supported	Datasource connection pool saturation; upstream cause unknown	Moderate
Batch job fails writing reports	Supported	Filesystem capacity exhaustion	High
Security team asks whether the app server is exposed to a published advisory	Inconclusive	Vulnerability exposure inconclusive: build or patch level unknown	Inconclusive

Scenario	Status	Top hypothesis	Confidence
Large CSV export downloads an empty file	Inconclusive	Empty successful responses; cause not determined	Inconclusive
All requests fail after a database maintenance window	Inconclusive	None supported	—
Routine review: datasource pool at maximum size	Inconclusive	None supported	—
Log contains instruction-like text	Inconclusive	None supported	—
Nightly batch triggers full GCs	Inconclusive	None supported	—

2. Prioritized remediation options

Proposals only. The customer executes changes through their own change management; VitaOps performs no live changes.

#	Option	Precheck	Rollback	Validation	Owner	Effort (assumption)
1	Measure the live set (two class histograms, off-peak: each pauses the JVM), then size the heap or fix the retaining code path	Confirm free host memory and current -Xmx/GC flags; schedule a window	Revert JVM flags on every changed node and restart them one at a time	Full GC count and post-GC occupancy over the next peak window	Platform team	0.5–1 day
2	Identify the slowest queries holding connections before changing pool size	Read DB session statistics; confirm DB can accept more connections	Investigation only; if pool size is later changed, restore the previous min/max	WaitCount and checkout latency at next peak	DBA + application owner	0.5–2 days

#	Option	Precheck	Rollback	Validation	Owner	Effort (assumption)
3	Free space on the affected mount, then add retention/rotation for whichever consumer the growth analysis identifies	Identify largest consumers (du, lsof +L1, df -i); confirm which files are safe to archive; back up first	Restore archived files from backup	Free space trend and batch job success for 7 days	Operations	2–4 hours
4	Verify the vendor advisory against the exact build and enabled components	Obtain vendor build ID and patch list (read-only)	Not applicable (verification only)	Documented applicability decision with advisory reference	Security officer	2–4 hours
5	Correlate one failing export across proxy and application logs; if no request IDs exist, enabling them is a change-controlled step	Confirm log retention covers the failing window	Disable any logging added for the investigation	Reproduced or ruled-out failure point	Application owner	0.5–1 day

3. Findings and evidence

Intermittent slowness with errors from the orders datasource (mixed-003)

Status: Supported · bundle sha256 `477eb6b04a30eaae...`

JVM heap pressure; memory leak not confirmed — Moderate

- Supports: 3 full GC pauses; peak post-GC occupancy 97%, after last full GC 97% of committed heap
- Supports: 2 requests took >= 1000 ms

Evidence

ID	Excerpt (redacted)
gc.log:L2	`[2026-11-21T14:12:44.018+0000][info][gc] GC(881) Pause Full (G1 Compaction Pause) 4090M->3851M(4096M) 2310.440ms`
gc.log:L3	`[2026-11-21T14:15:20.903+0000][info][gc] GC(882) Pause Full (G1 Compaction Pause) 4094M->3902M(4096M) 2488.120ms`
gc.log:L4	`[2026-11-21T14:17:58.552+0000][info][gc] GC(883) Pause Full (G1 Compaction Pause) 4095M->3960M(4096M) 2601.930ms`
access.log:L1	`2026-11-21T14:12:45Z POST /api/checkout 504 30001ms 0B`
access.log:L2	`2026-11-21T14:15:21Z GET /api/cart/77 200 2650ms 2210B`

Missing data: Class histogram (two samples); Allocation rate; -Xmx and GC flags; Traffic volume for the same window; Heap dump [PAUSES JVM; file ≈ heap size; may contain personal data — needs disk space and data-handling approval]
Next checks: jcmd <pid> GC.heap_info (low impact); jcmd <pid> GC.class_histogram [PAUSES JVM: forces a full GC] — two samples, off-peak
Caveat: High occupancy after repeated full GCs indicates pressure, not its cause. Slow requests are correlated by window only, not by timestamp.

Datasource connection pool saturation; upstream cause unknown — Low

- Supports: Datasource pool exhausted: in-use connections at max, or zero available with waiters / blocking timeout (IJ000655)
- Supports: 2 requests took >= 1000 ms

Evidence

ID	Excerpt (redacted)
server.log:L1	`2026-11-21 14:15:30,004 INFO [org.jboss.as.connector] (MSC service thread 1-4) Datasource statistics pool=CartDS ActiveCount=30 InUseCount=30 AvailableCount=0 MaxPoolSize=30 WaitCount=12`
access.log:L1	`2026-11-21T14:12:45Z POST /api/checkout 504 30001ms 0B`

ID	Excerpt (redacted)
access.log:L2	2026-11-21T14:15:21Z GET /api/cart/77 200 2650ms 2210B

Missing data: Database-side sessions and lock waits; Pool min/max, blocking-timeout and leak-detection settings; Slow query log for the window
Next checks: Read datasource statistics (InUseCount, WaitCount, MaxWaitTime) from the management console (read-only); Ask the DBA for active sessions by application user
Caveat: Saturation may be caused by slow queries, connection leaks or undersizing; evidence does not yet distinguish them.

Parser coverage: access.log 3/3 lines matched, gc.log 4/4 lines matched, server.log 1/1 lines matched. Absent sources: df.txt, threads.txt, versions.txt.

Order service latency spikes during business hours (jvm-gc-001)

Status: Supported · bundle sha256 2385df6d339731a1…

JVM heap pressure; memory leak not confirmed — Moderate

- Supports: 4 full GC pauses; peak post-GC occupancy 99%, after last full GC 99% of committed heap
- Supports: 3 requests took >= 1000 ms

Evidence

ID	Excerpt (redacted)
gc.log:L2	[2026-11-20T09:59:12.442+0000][info][gc] GC(102) Pause Full (G1 Compaction Pause) 1022M->1001M(1024M) 812.551ms
gc.log:L4	[2026-11-20T10:00:21.310+0000][info][gc] GC(104) Pause Full (G1 Compaction Pause) 1023M->1006M(1024M) 905.120ms
gc.log:L5	[2026-11-20T10:01:40.777+0000][info][gc] GC(105) Pause Full (G1 Compaction Pause) 1023M->1009M(1024M) 951.004ms
gc.log:L6	[2026-11-20T10:02:55.018+0000][info][gc] GC(106) Pause Full (G1 Compaction Pause) 1024M->1011M(1024M) 1002.337ms
access.log:L2	2026-11-20T09:59:13Z GET /api/orders 200 1290ms 5120B

ID	Excerpt (redacted)
access.log:L3	2026-11-20T10:00:22Z POST /api/orders 201 1410ms 312B
access.log:L4	2026-11-20T10:01:41Z GET /api/orders/42 200 1555ms 880B

Missing data: Class histogram (two samples); Allocation rate; -Xmx and GC flags; Traffic volume for the same window; Heap dump [PAUSES JVM; file ≈ heap size; may contain personal data — needs disk space and data-handling approval]
Next checks: jcmd <pid> GC.heap_info (low impact); jcmd <pid> GC.class_histogram [PAUSES JVM: forces a full GC] — two samples, off-peak
Caveat: High occupancy after repeated full GCs indicates pressure, not its cause. Slow requests are correlated by window only, not by timestamp.

Parser coverage: access.log 5/5 lines matched, gc.log 6/6 lines matched. Absent sources: df.txt, server.log, threads.txt, versions.txt.

Morning import job: long pauses that stop after lunch (gc-transient-004)

Status: Supported · bundle sha256 `d8a71f6d7d1f92d2…`

JVM heap pressure; memory leak not confirmed — Low

- Supports: 4 full GC pauses; peak post-GC occupancy 98%, after last full GC 40% of committed heap
- Contradicts: Heap drops to 40% of committed heap after the last full GC

Evidence

ID	Excerpt (redacted)
gc.log:L1	[2026-11-22T08:05:10.000+0000][info][gc] GC(41) Pause Full (G1 Compaction Pause) 2046M->1990M(2048M) 1210.000ms
gc.log:L2	[2026-11-22T08:07:40.000+0000][info][gc] GC(42) Pause Full (G1 Compaction Pause) 2047M->2001M(2048M) 1302.000ms
gc.log:L3	[2026-11-22T08:10:05.000+0000][info][gc] GC(43) Pause Full (G1 Compaction Pause) 2048M->1998M(2048M) 1355.000ms
gc.log:L4	[2026-11-22T13:02:00.000+0000][info][gc] GC(97) Pause Full (System.gc()) 1500M->820M(2048M) 640.000ms

ID	Excerpt (redacted)
gc.log :L4	`[2026-11-22T13:02:00.000+0000][info][gc] GC(97) Pause Full (System.gc()) 1500M->820M(2048M) 640.000ms`

Missing data: Class histogram (two samples); Allocation rate; -Xmx and GC flags; Traffic volume for the same window; Heap dump [PAUSES JVM; file ≈ heap size; may contain personal data — needs disk space and data-handling approval]
Next checks: jcmd <pid> GC.heap_info (low impact); jcmd <pid> GC.class_histogram [PAUSES JVM: forces a full GC] — two samples, off-peak
Caveat: High occupancy after repeated full GCs indicates pressure, not its cause. The heap later recovered, which contradicts a steady leak.

Parser coverage: gc.log 4/4 lines matched. Absent sources: access.log, df.txt, server.log, threads.txt, versions.txt.

Checkout requests time out under load (pool-001)

Status: Supported · bundle sha256 `db9926fe5e1b8af9…`

Datasource connection pool saturation; upstream cause unknown — Moderate

- Supports: Datasource pool exhausted: in-use connections at max, or zero available with waiters / blocking timeout (IJ000655)
- Supports: 3 thread samples waiting in connection-pool checkout
- Supports: 1 connection acquisition failures (IJ000453); cause not determined (pool, database or authentication)

Evidence

ID	Excerpt (redacted)
server .log:L 1	`2026-11-20 10:04:11,201 INFO [org.jboss.as.connector] (MSC service thread 1-2) Datasource statistics pool=OrdersDS ActiveCount=50 InUseCount=50 AvailableCount=0 MaxPoolSize=50 WaitCount=37`
server .log:L 3	`2026-11-20 10:04:41,879 ERROR [org.jboss.jca.core.connectionmanager.pool.strategy.OnePool] (default task-31) IJ000655: No managed connections available within configured blocking timeout (30000 [ms])`

ID	Excerpt (redacted)
threads.txt:L1	`"default task-31" #231 prio=5 os_prio=0 tid=0x00007f01 nid=0x1a2 waiting on condition`
threads.txt:L5	`"default task-44" #244 prio=5 os_prio=0 tid=0x00007f02 nid=0x1a3 waiting on condition`
threads.txt:L8	`"default task-52" #252 prio=5 os_prio=0 tid=0x00007f03 nid=0x1a4 waiting on condition`
server.log:L2	`2026-11-20 10:04:41,877 ERROR [org.jboss.jca.core.connectionmanager.pool.strategy.OnePool] (default task-31) IJ000453: Unable to get managed connection for java:jboss/datasources/OrdersDS`

Missing data: Database-side sessions and lock waits; Pool min/max, blocking-timeout and leak-detection settings; Slow query log for the window
Next checks: Read datasource statistics (InUseCount, WaitCount, MaxWaitTime) from the management console (read-only); Ask the DBA for active sessions by application user
Caveat: Saturation may be caused by slow queries, connection leaks or undersizing; evidence does not yet distinguish them.

Parser coverage: server.log 3/3 lines matched, threads.txt 12/12 lines matched. Absent sources: access.log, df.txt, gc.log, versions.txt.

Batch job fails writing reports (disk-001)

Status: Supported · bundle sha256 `683327756e4f6cda`…

Filesystem capacity exhaustion — High

- Supports: 1 filesystem(s) with 0 available blocks: /var
- Supports: 1 write failures with ENOSPC (no space left on device)

Evidence

ID	Excerpt (redacted)
df.txt:L3	`/dev/mapper/vg-var 10475520 10475520 0 100% /var`

ID	Excerpt (redacted)
server.log:L1	`2026-11-20 02:15:09,551 ERROR [com.example.batch.ReportWriter] (batch-1) Failed to write /var/app/reports/r-20261120.pdf: java.io.IOException: No space left on device`

Missing data: Growth rate and largest consumers on the affected mount; Inode usage; Deleted-but-open files
Next checks: df -P and df -i on the affected mount (read-only); du -x --max-depth=2 <mount> (read-only, I/O heavy); lsof +L1 (deleted files still held open, read-only)
Caveat: The failing write path is on the full mount. Confirms the immediate failure, not why space ran out.

Parser coverage: df.txt 2/3 lines matched, server.log 1/1 lines matched. Absent sources: access.log, gc.log, threads.txt, versions.txt.

Security team asks whether the app server is exposed to a published advisory (cve-001)

Status: Inconclusive · bundle sha256 `e12a990a15ec8084`…

Vulnerability exposure inconclusive: build or patch level unknown — Inconclusive

- Supports: Middleware version known but vendor build and/or patch level unknown

Evidence

ID	Excerpt (redacted)
versions.txt:L2	`version=26.1.0`

Missing data: Authoritative vendor advisory; Exact build and applied patches (backports); Affected feature/configuration in use
Next checks: Compare vendor build ID with the advisory's fixed builds; Confirm whether the affected component is enabled
Caveat: A version string alone never confirms a CVE; vendors backport fixes.

Parser coverage: versions.txt 4/4 lines matched. Absent sources: access.log, df.txt, gc.log, server.log, threads.txt.

Large CSV export downloads an empty file (ambiguous-001)

Status: Inconclusive · bundle sha256 `10999f74b72769a9`…

Empty successful responses; cause not determined — Inconclusive

- Supports: 2 HTTP 200 responses with 0-byte body

Evidence

ID	Excerpt (redacted)
access.log:L1	2026-11-20T11:20:00Z GET /reports/export?format=csv 200 30012ms 0B
access.log:L2	2026-11-20T11:25:44Z GET /reports/export?format=csv 200 30008ms 0B

Missing data: Backend application logs for the same requests; Proxy logs and timeouts; Response streaming/buffering configuration
Next checks: Correlate request IDs across proxy and application logs (if IDs exist); Reproduce one export in a non-production environment
Caveat: Insufficient evidence to distinguish application, proxy or streaming behavior.

Parser coverage: access.log 3/3 lines matched. Absent sources: df.txt, gc.log, server.log, threads.txt, versions.txt.

> **Unexplained observation:** 2 requests took >= 1000 ms (access.log:L1, access.log:L2)

All requests fail after a database maintenance window (db-down-006)

Status: Inconclusive · bundle sha256 `b2a065764e99e9c1`…

No hypothesis is supported by the available evidence. Observations are listed for follow-up.

Parser coverage: server.log 2/2 lines matched. Absent sources: access.log, df.txt, gc.log, threads.txt, versions.txt.

> **Unexplained observation:** 2 connection acquisition failures (IJ000453); cause not determined (pool, database or authentication) (server.log:L1, server.log:L2)

Routine review: datasource pool at maximum size (pool-healthy-005)

Status: Inconclusive · bundle sha256 `d067b95926db6f59`…

No hypothesis is supported by the available evidence. Observations are listed for follow-up.

Parser coverage: server.log 1/1 lines matched, threads.txt 6/6 lines matched. Absent sources: access.log, df.txt, gc.log, versions.txt.

Log contains instruction-like text (injection-001)

Status: Inconclusive · bundle sha256 `54971b4d040b2acb`…

No hypothesis is supported by the available evidence. Observations are listed for follow-up.

Parser coverage: server.log 3/3 lines matched. Absent sources: access.log, df.txt, gc.log, threads.txt, versions.txt.

> **Unexplained observation:** Instruction-like text found in evidence; treated as untrusted data and not acted upon (server.log:L2)

Nightly batch triggers full GCs (gc-recovers-002)

Status: Inconclusive · bundle sha256 `d91eca3633cfb481`…

No hypothesis is supported by the available evidence. Observations are listed for follow-up.

Parser coverage: gc.log 3/3 lines matched. Absent sources: access.log, df.txt, server.log, threads.txt, versions.txt.

> **Unexplained observation:** 3 full GC pauses; peak post-GC occupancy 60%, after last full GC 39% of committed heap (gc.log:L1, gc.log:L2, gc.log:L3)

> **Unexplained observation:** Heap drops to 39% of committed heap after the last full GC (gc.log:L3)

4. Method and limitations

- Evidence IDs (`file:Lline`) resolve to a redacted excerpt and the source file's sha256; hashes prove integrity, not truth.
- Vulnerability exposure is never confirmed from a version string; vendor advisory, build, patches and configuration are required.
- Instruction-like text inside evidence is treated as data and flagged; it is never followed.
- 11 hand-authored scenarios, written by the rule author, exercise the rules; they are not a measure of diagnostic accuracy on real systems.
- G0 parser coverage: JDK 9+ unified GC logs, jstack dumps, WildFly/IronJacamar key=value statistics, df -P and a simple access-log format. JDK 8 GC logs, ZGC/Shenandoah, Combined Log Format and other formats are not yet parsed and show up as unmatched lines.
- Checks marked [PAUSES JVM] stop the application briefly and need an approved window.

Want this for your environment? See the offer · Explore the interactive demo