



Vantage RuntimeAI

Batch agent check-rides, model comparison, and operational scorecards for autonomous AI workflows

8.0/10

Overall score

Run summary

Scenario: Freshness SLA Breach

Model: ibm-granite/granite-4.0-h-micro

Roles: Data Engineer (AI agent) <> Engineering counterpart (AI agent)

Duration: 1m 39s

Actual run cost: \$0.000259

Projected execution cost: \$0.2847/mo (50/biz day × 22 days)

Report date: July 13, 2026 - 10:35 PM EST

Overall

8.0/10

PASS - trust=high

Pass - score 8.0/10 clears 7.0 with acceptable trust/closure.

CAPABILITY RUBRIC



Spoke numbers run clockwise from the top; dimension names are listed in full in the table.

Technical criteria checklist

Automated pass/fail signals for this task - use when comparing models on specific requirements.

- Upstream blocker identified: Met

- SLA breach acknowledged: Met

- Downstream impact noted: Met

- Restore steps listed: Met

TECHNICAL JUDGEMENT

Holistic read on this single-shot deliverable - criteria checklist plus rubric gaps, not any one dimension alone.

8/10

Telemetry

Run metrics

Mode: multi-turn - Agent msgs: 4 - Counterpart msgs: 5 - Output: 12305 chars - Latency: 99s - Criteria: 4/4

Score analysis

Identifies int_order_enriched failure as upstream blocker.

Score snapshot

Solid: Solid technical task response with a few gaps.

Interpretation

Multi-turn technical task: fixed brief, 4 agent turns with scripted follow-ups and reviewer sign-off. 4 of 4 checklist criteria passed.

BRIGHT SPOTS (RUBRIC)

- + Stale upstream identified: Correctly identifies the primary technical issue from the artifacts.
- + Uses SLA / freshness facts: Uses tables, logs, or schema facts from the task brief.
- + Downstream blast radius: Shows appropriate safety, budget, or risk awareness for the scenario.
- + Structured status summary: Clear structure with readable SQL, diff, or memo formatting.
- + Restore & verify steps: Strong rationale and actionable follow-through.

DIMENSION NOTES

- Stale upstream identified: Correctly identifies the primary technical issue from the artifacts.
- Uses SLA / freshness facts: Uses tables, logs, or schema facts from the task brief.
- Downstream blast radius: Shows appropriate safety, budget, or risk awareness for the scenario.
- Structured status summary: Clear structure with readable SQL, diff, or memo formatting.
- Restore & verify steps: Strong rationale and actionable follow-through.
- Overall judgement (8/10): Sound judgement on this technical response: checklist criteria and rubric align with a strong deliverable.

Signals

Upstream blocker identified: yes - SLA breach acknowledged: yes - Downstream impact noted: yes - Restore steps listed: yes

CAVEATS

- Heuristic keyword/signal scoring - use the criteria grid and transcript to validate edge cases.

Transcript

Session

Scenario: Freshness SLA Breach

Agent response <-> Engineering counterpart

Data Engineer (AI agent) <> Engineering counterpart (AI agent)

Duration: 1m 39s

Engineering counterpart

2:33 AM

Task brief - Freshness SLA breach (moderate complexity)

A P1 mart breached its freshness SLA. Triage using the metadata JSON:

```
```json
{
 "mart": "analytics.fct_orders",
 "owner_team": "data-platform",
 "sla": {
 "max_staleness_minutes": 120,
 "tier": "P1 revenue dashboard"
 },
 "freshness": {
 "last_successful_run_utc": "2026-07-05T09:12:00Z",
 "now_utc": "2026-07-05T15:40:00Z",
 "staleness_minutes": 388,
 "status": "BREACHED"
 },
 "upstream": [
 {"model": "staging.stg_orders", "last_run_utc": "2026-07-05T15:05:00Z", "status": "success"},
 {"model": "staging.stg_customers", "last_run_utc": "2026-07-05T14:58:00Z", "status": "success"},
 {"model": "intermediate.int_order_enriched", "last_run_utc": "2026-07-05T09:12:00Z", "status": "failed", "error":
"DBT_RUN_TIMEOUT after 45m"}
],
 "downstream_consumers": ["Looker Revenue Overview", "exec_daily_orders Slack bot"],
 "note": "int_order_enriched has been failing since morning; fct_orders has not refreshed."
}
```
```

Agent response

2:34 AM - Responded in 24s

Freshness SLA Breach Analysis & Remediation Plan

1. Snapshot of the Breach

| Item | Value |
|------------------------|--|
| Mart | `analytics.fct_orders` |
| Owner Team | `data-platform` |
| SLA | Max staleness 120?min, Tier P1 revenue dashboard |
| Current Staleness | 388?min (6?h?28?min) |
| Last Successful Run | `2026-07-05T09:12:00Z` |
| Current Time | `2026-07-05T15:40:00Z` |
| Blocked Upstream Model | `intermediate.int_order_enriched` (failed with `DBT_RUN_TIMEOUT`) |
| Downstream Consumers | `Looker Revenue Overview`, `exec_daily_orders Slack bot` |
| Note | `int_order_enriched` has been failing since morning; `fct_orders` has not refreshed. |

Bottom line: The failure of `int\_order\_enriched` caused a cascade that prevented `fct\_orders` from being refreshed, resulting in a P1 breach well beyond the 120?minute SLA.

2. Root Cause Diagnosis

| Step | Observation | Reasoning |
|------|---------------------|---|
| a. | Upstream Dependency | `int_order_enriched` last run failed at `09:12?UTC` (DBT timeout). DBT timeout indicates the model could not complete within its allocated time (45?min). |
| b. | Dependency Chain | `int_order_enriched` -> `staging.stg_orders` -> `staging.stg_customers` -> `analytics.fct_orders`. Each downstream model depends on the successful completion of its upstream dependencies. |
| c. | Time Gap | `int_order_enriched` failed before `fct_orders` could be refreshed. The failure created a blocking upstream condition that halted the entire pipeline. |
| d. | SLA Impact | Staleness grew to 388?min (far beyond 120?min). The breach is P1 because it directly impacts a revenue?critical dashboard. |

Conclusion: The blocking upstream model is `intermediate.int\_order\_enriched`. Its failure caused the entire downstream refresh to stall, creating the SLA breach.

3. Blast Radius Assessment

| Downstream Consumer | Affected |
|-------------------------------------|---|
| Looker Revenue Overview | High impact - real?time revenue metrics will be stale, potentially misleading business decisions. |
| exec_daily_orders Slack bot | Medium impact - daily order summary will be outdated; may affect executive reporting. |
| Other downstream consumers (if any) | Potential impact - any consumer relying on `fct_orders` for fresh data will experience delays. |

Blast radius: Medium?high - the breach directly affects two critical dashboards and any downstream processes that consume `fct\_orders`.

4. Ordered Remediation Steps (to restore mart within SLA)

> Goal: Bring `fct\_orders` back to a fresh state (??120?min staleness) within the next 2?hours (to stay within the P1 SLA window).

Step?by?Step Plan

| # | Action | Owner | Estimated Duration | Dependencies |
|---|--|-------|--------------------|--------------|
| 1 | Trigger a manual refresh of `int_order_enrich` | | | |

Thanks - walk me through your reasoning step by step. What assumptions did you make?

Agent response

2:34 AM - Responded in 26s

Below is a step-by-step walk-through of how I arrived at the diagnosis, blast-radius assessment, and remediation plan for the "Freshness SLA breach" you described. I also explicitly list the assumptions that underlie each step so you can see where the reasoning is anchored.

1?? Gather the Raw Facts (Metadata JSON)

```
Field	Value	Why it matters
mart	`analytics.fct_orders`	This is the target mart that is supposed to be refreshed.
owner_team	`data?platform`	The team responsible for the mart - we'll involve them in the fix.
sla	`max_staleness_minutes: 120`, tier: P1 revenue dashboard	The SLA is P1 (critical) and the allowed staleness is 120?min. Anything beyond that is a breach.
freshness	`last_successful_run_utc: 2026?07?05T09:12:00Z`   `now_utc: 2026?07?05T15:40:00Z`   `staleness_minutes: 388`   `status: BREACHED`	Staleness = 388?min -> far beyond the 120?min SLA. The breach is P1 because it impacts a revenue?critical dashboard.
upstream	List of three models: 1. `staging.stg_orders` - success 2. `staging.stg_customers` - success 3. `intermediate.int_order_enriched` - failed (`DBT_RUN_TIMEOUT` after 45?min`)	The failure is upstream of `fct_orders`. The failure model is the blocking upstream.
downstream_consumers	`Looker Revenue Overview`, `exec_daily_orders Slack bot`	These are the consumers that will be affected by the outage.
note	`int_order_enriched` has been failing since morning; `fct_orders` has not refreshed.`	Confirms that the failure is ongoing and that `fct_orders` never got refreshed.
```

Assumptions Made Here

1. Metadata is accurate - The JSON reflects the current state of the data?pipeline. No hidden "silent" failures or missing fields.
2. Time zones are consistent - All timestamps are in UTC, so the 388?minute gap is measured correctly.
3. Only one upstream model is failing - The JSON lists three models; only `int\_order\_enriched` shows a failure. We assume no other upstream model is blocking.
4. All downstream consumers are directly dependent on `fct\_orders` - No other intermediate mart is feeding them; otherwise the blast radius would be larger.

2?? Diagnose the Root Cause

2.1 Identify the blocking upstream model

- Step: Scan the `upstream` array for any entry whose `status` is not "success".
- Result: `intermediate.int\_order\_enriched` -> failed with `DBT\_RUN\_TIMEOUT`.
- Why it blocks: In most ELT/ELT pipelines, a downstream model (`fct\_orders`) will wait for its upstream dependencies to finish. If an upstream model times out, the downstream job cannot start, causing a pipeline stall.

2.2 Trace the Dependency Chain

...

int_order_enriched -> staging.stg_orders -> staging.stg_customers

Engineering counterpart

2:34 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

2:35 AM - Responded in 24s

Below is a checklist of verification items you should run through before you promote the fix (or any change) from a dev/test environment to production. I've grouped them into logical buckets and added a short rationale for each so you can see why it matters for a "Freshness SLA breach" scenario.

1?? Pre?Production Validation Checklist

| # | Verification Area | What to Check | Why It Matters for SLA Breaches |

|---|-----|-----|-----|

| 1 | Data?model correctness | - Verify that `int\_order\_enriched` now runs successfully (no timeout, no errors).
- Confirm that the model's output schema matches the downstream expectations (`staging.stg\_orders`). | A broken model can cause downstream jobs to stall, repeat the SLA breach, or produce wrong data. |

| 2 | Run?time configuration | - Confirm that the DBT (or other orchestration) execution timeout is set appropriately (e.g., 60?min) and that the environment variables (e.g., `DBT\_CONFIG\_\_RUN\_TIMEOUT`) are correct.
- Verify that any resource limits (CPU, memory) are sufficient for the enriched data volume. | If the timeout is too low, the model will still fail; if resources are insufficient, the job may time?out even when the data is fine. |

| 3 | Dependency health | - Run a dry?run of the full pipeline (or at least the chain `int\_order\_enriched -> staging.stg\_orders -> staging.stg\_customers`).
- Check that each upstream model finishes within its SLA (e.g., `staging.stg\_orders` ??30?min, `staging.stg\_customers` ??15?min). | Guarantees that the fix does not introduce a new blocker that could cascade again. |

| 4 | SLA?aware monitoring | - Add or verify SLA?monitoring alerts for `fct\_orders` (e.g., "staleness > 120?min").
- Ensure alerts fire immediately (e.g., via Prometheus + Alertmanager, Datadog, CloudWatch). | Early detection prevents a breach from becoming a P1 incident. |

| 5 | Back?fill / re?process safety | - Test a re?run of `int\_order\_enriched` back?filling any missed rows (e.g., using `dbt run --full-refresh` or a scheduled incremental refresh).
- Verify that downstream consumers (Looker, Slack bot) receive consistent data after the refresh. | Guarantees that once the model runs, the downstream dashboards will be refreshed within the required window. |

| 6 | Performance & resource profiling | - Run a profiling job (e.g., `dbt run --profile`) to capture query execution times, I/O, and any bottlenecks.
- Compare against baseline (pre?fix) performance metrics. | Helps you prove that the fix actually reduces the run time, thus giving you a buffer to meet the 120?min SLA. |

| 7 | Rollback / fallback plan | - Document the exact steps to revert to the previous version of `int\_order\_enriched` (e.g., keep the previous DBT version tag, keep a snapshot of the staging tables).
- Ensure the rollback can be executed within minutes. | Production changes must be recoverable; a single mis?step

Engineering counterpart

2:35 AM - Responded in 0s

What are the top risks, and what's the rollback plan if this doesn't work?

Agent response

2:35 AM - Responded in 24s

Below is a concise risk assessment matrix followed by a step-by-step rollback plan you can execute (or have on-call run) if the fix above does not bring `fct\_orders` back to a fresh state within the SLA window.

1?? Top Risks (Why They Matter)

| # | Risk | Likelihood (H/M/L) | Impact (H/M/L) | Why It's Critical for a P1 SLA |
|---|------|--------------------|----------------|--------------------------------|
|---|------|--------------------|----------------|--------------------------------|

| | | | | |
|-----|-------|-------|-------|-------|
| --- | ----- | ----- | ----- | ----- |
|-----|-------|-------|-------|-------|

| | | | | |
|----|---|---|---|---|
| R1 | Model failure recurrence - `int_order_enriched` still times out or produces corrupted data. | M | H | If the underlying cause (e.g., a bad query, missing index, or resource contention) isn't fixed, the next refresh will again stall, creating another SLA breach. |
|----|---|---|---|---|

| | | | | |
|----|--|---|---|--|
| R2 | Dependency chain breakage - Even after `int_order_enriched` succeeds, downstream models (`staging.stg_orders`, `staging.stg_customers`) may still be stuck because they depend on the refreshed `int_order_enriched` tables. | M | H | A downstream job that cannot read the new enriched data will also stall, propagating the breach downstream (e.g., Looker dashboard). |
|----|--|---|---|--|

| | | | | |
|----|---|---|---|--|
| R3 | Resource contention / capacity - The environment (DBT workers, warehouse slots) is already near capacity; the new run may be throttled, causing a delay that pushes staleness beyond 120 min. | L | H | If the pipeline runs slower than expected, the 120 minute window could be missed, especially during peak load periods. |
|----|---|---|---|--|

| | | | | |
|----|---|---|---|--|
| R4 | Configuration drift - The DBT config (run timeout, resource limits) was changed in dev but not propagated to prod, or the environment variables differ. | L | M | A mismatch can cause the same timeout error even though the underlying data is fine. |
|----|---|---|---|--|

| | | | | |
|----|--|---|---|--|
| R5 | Monitoring blind spot - New alerts fire but are ignored (e.g., alert thresholds set too high, or the alert channel is down). | L | M | Without timely notification, the incident may go unnoticed until it becomes a full breach. |
|----|--|---|---|--|

| | | | | |
|----|--|---|---|--|
| R6 | Rollback failure - The rollback steps themselves cause a new outage (e.g., trying to revert a schema change that other teams rely on). | L | H | A rollback itself can become a new incident, especially if you revert a schema change that other downstream models also depend on. |
|----|--|---|---|--|

| | | | | |
|----|---|---|---|---|
| R7 | Data quality / integrity issues - The enriched data may contain missing or malformed rows that downstream consumers cannot handle gracefully. | M | M | Bad data can cause downstream dashboards to show "null" or "error" values, masking the true SLA breach. |
|----|---|---|---|---|

> Bottom line: The single biggest risk is R1 - the upstream model not actually fixing the root cause. All other risks are downstream consequences of that failure.

2?? Rollback Plan (If the Fix Fails)

> Goal: Restore the system to a known good state within minutes so that the SLA breach is halted and you can investigate the root cause without further impact.

2.1 Preconditions (Before You Start)

| Item | Action | Owner |
|------|--------|-------|
|------|--------|-------|

| | | |
|-------|-------|-------|
| ----- | ----- | ----- |
|-------|-------|-------|

| | | |
|---------------------------|---|---------------|
| Snapshot of current state | Capture the current `fct_orders` table snapshot (e.g., a `COPY TO` of the latest rows or a backup of the underlying source tables). | Data Engineer |
|---------------------------|---|---------------|

| | | |
|----------------------|---|--|
| Rollback version tag | Tag the DBT version used in the failed run (e.g., `dbt run --target prod -- | |
|----------------------|---|--|

Engineering counterpart**2:35 AM - Responded in 0s**

Thanks - upstream blocker and blast radius are well ordered. Prioritize revenue-facing dashboards if full freshness slips past the hour mark. Review complete - no further response needed.