



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: amazon/nova-micro-v1

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

Duration: 21s

Actual run cost: \$0.000406

Projected execution cost: \$0.4466/mo (50/biz day × 22 days)

Report date: July 14, 2026 - 2:52 AM EST

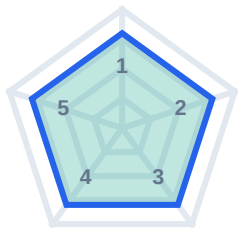
Overall

8.0/10

PASS - trust=high

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

CAPABILITY RUBRIC



#	Dimension (matches spoke #)	Score
1	Stale upstream identified	8/10
2	Uses SLA / freshness facts	8/10
3	Downstream blast radius	8/10
4	Structured status summary	8/10
5	Restore & verify steps	8/10

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: 12936 chars - Latency: 21s - 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: 21s

Engineering counterpart

6:52 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

6:52 AM - Responded in 4s

Task Assessment: Freshness SLA Breach in analytics.fct_orders

1. Key Metrics and SLA Breach:

- Mart in Question: `analytics.fct\_orders`
- Freshness SLA: Max staleness within 120 minutes (P1 tier, revenue dashboard)
- Freshness Status:
 - Last successful run: 2026-07-05T09:12:00Z
 - Current time: 2026-07-05T15:40:00Z
 - Staleness: 388 minutes (well over the 120-minute SLA)

2. Identifying the Blocking Upstream Model:

- Upstream Models:
 - `staging.stg\_orders`: last run success at 2026-07-05T15:05:00Z
 - `staging.stg\_customers`: last run success at 2026-07-05T14:58:00Z
 - `intermediate.int\_order\_enriched`:
 - Last run success: 2026-07-05T09:12:00Z
 - Status: failed
 - Error: `DBT\_RUN\_TIMEOUT` after 45m`
- Blocking Model: `intermediate.int\_order\_enriched`
 - It has failed, causing `analytics.fct\_orders` not to refresh.

3. Blast Radius on Downstream Consumers:

- Downstream Consumers:
 - `Looker Revenue Overview`
 - `exec\_daily\_orders` Slack bot`
- Impact: Both downstream consumers rely on `analytics.fct\_orders`, hence are impacted by the freshness SLA breach.

Steps to Restore the Mart within SLA:

Step 1: Troubleshoot the Intermediate Model Failure

1. Investigate the Failure Cause:

- Check DBT run logs for `int\_order\_enriched`.
- Review recent changes in code/data sources for potential issues.

2. Address the DBT Timeout Issue:

- Review DBT configuration files for any resource constraints (execution time, memory, etc.).
- Optimize the model query or add necessary timeout settings.

3. Manually Rerun the Intermediate Model:

- After addressing the issue, attempt a manual rerun to ensure it completes successfully.
- Example command: `dbt run --select int\_order\_enriched`

Step 2: Refresh the Fact Model (`analytics.fct\_orders`)

1. Once the Intermediate Model is Successful:

- Run the fact model manually using DBT to ensure data flow.
- Example command: `dbt run --select fct\_orders`

Step 3: Validate Data and SLA Compliance

1. Post-refresh validation:

- Check data consistency between `int\_order\_enriched` and `analytics.fct\_orders`.
- Validate the freshness by checking the last run time of `analytics.fct\_orders`.
- Ensure downstream systems (`Looker`, `Slack bot`) are pulling updated data.

Step 4: Monitor and Further Prevent SLA Breaches

1. Monitor closely for any intermittent issues:

- Set up alert thresholds based on SLA.
- Ensure there is always a monitoring check in place post immediate resolution.

2. Optimization and Capacity Planning:

- Regularly review and optimize DBT runs for performance improvements.
- Increase execution timeouts if consistently hitting limits.
- Consider scaling up resources if DBT runs are regularly hitting execution limits.

3. Document the

Engineering counterpart
6:52 AM - Responded in 0s

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

Agent response

6:52 AM - Responded in 4s

Sure, let's walk through the reasoning step-by-step in detail, including the assumptions made at each stage.

1. Identify the Blocking Upstream Model:

- Observation: In the metadata JSON provided, the `analytics.fct\_orders` mart has not refreshed since the `intermediate.int\_order\_enriched` model failed.
- Blocking Model: The `intermediate.int\_order\_enriched` model is clearly marked as `failed` with a `DBT\_RUN\_TIMEOUT` error.
- Assumption: Since this model directly feeds into `analytics.fct\_orders`, its failure directly prevents the mart from refreshing.

2. Determine Impact on Downstream Consumers:

- Observation: The downstream consumers listed are `Looker Revenue Overview` and `exec\_daily\_orders Slack bot`.
- Impact: Both consumers rely on the `analytics.fct\_orders` mart. Since it has not refreshed properly, these downstream systems will feed stale or incomplete data.
- Assumption: If `analytics.fct\_orders` is not refreshing correctly, downstream systems will continue to suffer data delays.

3. Prioritize Steps to Restore Mart within SLA:

Step 1: Troubleshoot and Resolve Intermediate Model Issue

- Immediate Problem: The `DBT\_RUN\_TIMEOUT` indicates the model was taking too long to execute.
- Action: Investigate and resolve why the model execution exceeded its timeout limit.
 - Assumption: Timeouts often stem from performance bottlenecks or resource constraints.
- Example Troubleshooting Steps:
 - Check the execution time of the model against the timeout limit.
 - Review recent changes for optimization potential, like query adjustments.
 - Look into resource allocation (memory, CPU) during execution.
 - Review and possibly update DBT configuration settings like execution timeouts.

Step 2: Manually Rerun Intermediate Model

- Action: Rerun the `intermediate.int\_order\_enriched` model manually after resolving the underlying issue.
- Assumption: Fixing the issue will allow the intermediate model to run successfully.

Step 3: Refresh Fact Model (`analytics.fct\_orders`)

- Action: Once the intermediate model is confirmed as successful, rerun the fact model.
 - Assumption: Successful completion of the intermediate model should allow `analytics.fct\_orders` to refresh without issues.

Step 4: Validate Data and SLA Compliance

- Action: Post-refresh, validate that the data consistency is correct and the mart has indeed refreshed within the SLA.
- Assumption: Post-refresh validation ensures the SLA breach has been fully resolved.

Step 5: Monitor to Prevent Future Breaches:

- Action: Set up monitoring alerts and review performance regularly.
 - Assumption: Regular monitoring will catch any issues early and allow preemptive actions to avoid future SLA breaches.

Additional Assumptions:

- Operational Assumption: That the team has access to DBT logs, system performance metrics, and the ability to run manual commands on the relevant models.
- Technological Assumption: That the underlying database and processing environment is healthy and can support these operations without additional constraints.

By following these steps, the immediate breach is addressed while also taking proactive measures to prevent future occurrences. This approach focuses on the root cause of the SLA breach (the failing intermediate model) and ensures the restoration of data freshness for downstream consumers.

Engineering counterpart
6:52 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

6:52 AM - Responded in 4s

Before proceeding to ship this fix and the renewed data model to production, it's important to ensure everything works smoothly and meets the required standards. Here's a comprehensive checklist to verify before moving forward:

1. Model Validation

- Successful Execution:

- Ensure ``intermediate.int_order_enriched`` successfully completes without the previous timeout error.
- Verify the output schema and data types match expectations.
- Confirm the data volume post-execution as expected.

- Data Integrity:

- Check for data consistency and correctness against known good outputs or source data.
- Validate data quality standards are met, no missing or erroneous data.

2. Performance Testing

- Execution Time:

- Confirm that ``intermediate.int_order_enriched`` executes within acceptable time limits and the timeout issue is resolved.

- Resource Utilization:

- Monitor CPU, memory, and database usage to ensure the model does not overload the system.

- Throughput:

- Verify that the model can handle expected load and future scaling requirements.

3. Integration Testing

- Upstream to Downstream Flow:

- After successfully running ``intermediate.int_order_enriched``, manually trigger ``analytics.fct_orders`` and verify its successful execution.
- Ensure there are no errors reported in the DBT logs and that data makes it from the intermediate model to the fact model.

- Data Validation Post-Integration:

- Validate the ``analytics.fct_orders`` output again for consistency, completeness, and correctness.

4. End-to-End Testing

- Downstream Impact:

- Confirm data feeds into downstream consumers (Looker & Slack bot) are updated with fresh data.
- Perform manual or automated checks to ensure dashboards in Looker and any downstream reports/summaries reflect the updated data.

- Business Consistency:

- Validate key business metrics and reports are now reflecting the latest data accurately.

5. Documentation and Communication

- Change Documentation:

- Document changes made to resolve the timeout issue.
- Include details of performance tuning, model adjustments, and any new configurations.

- Stakeholder Communication:

- Inform stakeholders that the issue is resolved and data is expected to be correctly reflected downstream.
- Prepare for any potential questions or additional issues.

6. Monitoring Setup

- Alerts and Monitoring:

- Ensure that monitoring alerts are set up to notify the team of any future failures or performance degradations.
- Confirm that logs are archived and accessible for review, should there be an issue post-deployment.

7. Code Review and Approval

- Peer Review:
 - Conduct a code review to ensure all changes meet team standards.
 - Obtain necessary approvals from team leads or relevant stakeholders prior to deployment.

By thoroughly verifying each of the above steps, you help ensure that the fix is reliable, data accuracy is restored, and the system is prepared to maintain these standards in production.

Engineering counterpart
6:52 AM - Responded in 0s

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

Agent response

6:52 AM - Responded in 9s

When deploying fixes for a systems breach, particularly in a production environment, understanding the top risks and having a rollback plan is crucial. Here's a summary of the top risks, along with a rollback plan for each, should the fix not work as expected:

Top Risks

1. Data Inconsistency

- Risk: Data may not propagate correctly from the intermediate model to the final mart, leading to inconsistent or stale data in downstream systems.
- Rollback Plan: Temporarily reroute data flows to use a previously successful snapshot or restore data from a reliable backup. Validate that the rollback data is coherent and then gradually phase back into the new data streams.

2. Performance Degradation

- Risk: Changes might lead to slower performance or resource contention, impacting the system's ability to handle regular operations.
- Rollback Plan: Identify the point of failure, revert configuration changes or code to previous versions if necessary, and monitor system performance to ensure it returns to acceptable levels.

3. Downtime

- Risk: Deployment could result in system downtime, especially if the models are re-run manually or if there's an outage during the integration process.
- Rollback Plan: Prepare for an immediate fallback to a stable version of the systems. Utilize staging or test environments to make adjustments without affecting production.

4. Security Vulnerabilities

- Risk: Changes in configurations or code might inadvertently introduce security vulnerabilities or affect access control.
- Rollback Plan: Implement immediate monitoring of security logs for any abnormal activities. Revert any changes that might affect system security and engage security experts if needed.

5. Business Impact

- Risk: Downstream systems relying on the data might not receive timely or correct information, affecting business operations.
- Rollback Plan: Communicate with business stakeholders to manage expectations. Implement a temporary solution to keep business processes running, such as using cached data or manual updates where feasible.

General Rollback Plan

- Immediate Suspension of Changes: Halt any ongoing deployments or changes to prevent further issues.
- System Restoration: Based on the risk identified, revert to the last known good state of the system. This might include restoring to a previous version of the data model, switching back to the old codebase, or reverting any configuration changes.
- Communication: Inform all stakeholders about the situation and the steps being taken to resolve it. Provide regular updates on the situation and the timeline for resolution.
- Investigation: Once stabilized, conduct a detailed investigation to determine the root cause of the failure and avoid similar issues in the future. Documentation of this process is vital for learning and improvement.
- Testing and Verification: Before re-attempting the solution, conduct thorough testing in a non-production environment to verify fixes will succeed without introducing new issues.
- Redeployment: Only after thorough testing and stakeholder approval, redeploy the changes. Continuously monitor systems for any signs of failure post-deployment.

In any rollback plan, communication and documentation are key. This ensures that all stakeholders are informed and that the rollback process itself does not lead to further complications.

Engineering counterpart

6:52 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.