AI Automation / Task Orchestration

Define metrics and alerts for workflow health — latency, failure rates, throughput, queue depth — detects problems before users complain.
Difficulty: Intermediate
Model: GPT-4 / Claude / Gemini
Use Case: Monitoring, Alerting, Observability
Updated: May 2026
Why This Prompt Exists
Workflows fail. The problem isn’t failure — it’s not knowing until users complain. Monitoring turns silent failures into actionable alerts.

You get:

- workflows that have been failing for days (no one noticed)
- slow degradation with no alert (users frustrated, then leave)
- capacity issues that cause queue buildup (no visibility)
- alert fatigue (too many false alarms, real issues ignored)
- no baseline for “normal” behavior (can’t detect anomalies)

But monitoring has patterns:

- latency: how long does each workflow take?
- throughput: how many workflows complete per minute/hour?
- failure rate: what percentage of runs fail?
- queue depth: how many workflows are waiting?
- error types: what kinds of failures are happening?

Without monitoring, you’re flying blind.

This prompt designs comprehensive workflow monitoring.

The Prompt

```
Assume the role of an observability engineer who designs workflow
monitoring.

Your task is to define metrics and alerts for a workflow.

Generate:

1. KEY METRICS TO TRACK

| Metric | Description | Target | Measurement |
|--------|-------------|--------|-------------|
| Latency (p95) | Time from trigger to completion | < 5s | Workflow
duration |
| Throughput | Runs per minute | 100/min | Counter |
| Failure rate | % of runs that error | < 1% | Error counter / total |
| Queue depth | Waiting workflows | < 10 | Queue length |
| Age of oldest item | Time oldest waiting | < 60s | Timestamp diff |

2. ALERT RULES

| Metric | Condition | Severity | Action |
|--------|-----------|----------|--------|
| Latency | > 10s for 5 min | Warning | Slack #alerts |
| Failure rate | > 5% for 2 min | Critical | PagerDuty |
| Queue depth | > 100 for 10 min | Warning | Investigate capacity |
```

3. DASHBOARD RECOMMENDATIONS
   - Real-time view: [what metrics to show]
   - Historical trends: [last hour, day, week]
   - Per-task breakdown: [which tasks are slow/failing]

4. LOGGING REQUIREMENTS
   - What to log: [workflow ID, start time, end time, outcome, error details]
   - Log retention: [30 days / 90 days / 1 year]
   - Sampling: [100% for errors, 1% for successes]

5. ON-CALL RESPONSE
   - Critical alert → page on-call
   - Warning alert → Slack notification, handle during business hours

6. BASELINE ESTABLISHMENT
   - Run for [X] days to establish normal ranges
   - Expected values: [latency, throughput, failure rate]

INPUTS:

Workflow description (from previous TO prompts):
[E.G., "Customer order processing workflow"]

Expected volume:
[E.G., "10,000 orders per day"]

Criticality:
[MISSION-CRITICAL / IMPORTANT / NICE-TO-HAVE]

```
Existing monitoring tools:
[E.G., "DataDog, Slack, PagerDuty"]

RULES:
- Start with latency (p95), failure rate, and throughput — the three essentials
- Set alert thresholds above normal variation (avoid false alarms)
- Use p95 for latency (ignore extreme outliers, focus on typical user)
- Log enough detail to debug (workflow ID, input summary, error stack)
- Review and adjust thresholds monthly (baselines change as systems evolve)
- Test alerts by triggering failures intentionally
```

How To Use It

- Start with latency (p95), failure rate, and throughput — the three essentials.
- Set alert thresholds above normal variation — avoid false alarms.
- Use p95 for latency (ignore extreme outliers, focus on typical user).
- Log enough detail to debug (workflow ID, input summary, error stack).
- Review and adjust thresholds monthly — baselines change as systems evolve.
- Test alerts by triggering failures intentionally.

Example Input

Workflow description:
“Customer order processing workflow — from payment to fulfillment”

Expected volume:
“50,000 orders per day (peak: 10,000/hour)”

Criticality:

“MISSION-CRITICAL — revenue impact if down”

Existing monitoring tools:
“DataDog, Slack, PagerDuty”

Why It Works
Most workflows are deployed with no monitoring — the first sign of trouble is a user complaint. By then, it’s often too late.

This framework improves outcomes by forcing:

- metric definition (what to measure?)
- alert rule specification (when to notify?)
- dashboard design (what to show in real time?)
- logging requirements (what to save for debugging?)
- on-call response (who gets paged for which alerts?)

Failure modes this prevents:

- Silent failures — workflow broken for days, no one knows
- Queue backup — 10,000 workflows waiting, no alert
- Alert fatigue — 100 false alarms per day, real issues missed
- No debug data — can’t figure out why workflow failed

This improves on: “Deploy and pray” operations. Monitoring is not optional for production workflows.

Related to: TO-04 (Recovery) for handling failures; TO-05 (Human-in-Loop) for escalation paths.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also Parallel Execution Designer