AI Agent Evaluation Launch Checklist

A practical enterprise checklist for Copilot Studio, RAG, document AI, and workflow agents before production

Yangming Li
Applied AI Engineer · LLM Evaluation · RAG · Data Products
yangmingli.com

v0.1 · Lead Magnet Draft

Purpose

Most AI agents look impressive in demos. This checklist helps teams decide whether an agent is actually ready to ship: not just by judging answers, but by testing tasks, traces, tools, graders, release gates, and live monitoring loops.

How to use this checklist

This is not a benchmark cookbook. It is a practical operating model for turning an AI demo into a reviewable production system. Start small, use real failures, instrument traces, separate capability learning from regression protection, and define release gates before the agent can affect users or business records.

One-sentence version

Real failures + isolated harness + repeated trials + calibrated graders + trace review + capability/regression separation + release gates + online monitoring.

The 30-7-3-4-1 implementation plan

HeaderBlue

30 tasks	Collect 20–50 real failures, internal-test issues, or high-impact user scenarios; select the first 30 tasks by user impact.	Do not start with 500 synthetic benchmark tasks.
7 days	Run the harness end-to-end for one week with clean state, isolated permissions, repeated trials, and full trace capture.	Treat traces as logs, not temporary debugging files.
3 grader families	Use rules/programmatic checks, LLM judges, and human review samples. Start coarse, then specialize.	Avoid one generic “quality” score.
4 release gates	Regression suite, end-to-end success threshold, cost/latency budget, anomaly circuit breaker.	No release without stop lines.
1 online loop	Monitor success rate, retry rate, cost, failure categories, drift, and feed weekly top failures back into the eval set.	Production feedback becomes the dataset flywheel.

1 Why agent evaluation is hard

- Agent workflows include tool calls, retrieval, permissions, external systems, multiple steps, and changing environment state.
- A final answer can look successful while the trace reveals a wrong tool, stale document, hidden retry loop, or incorrect state change.
- Offline evaluation can diverge from production reality when APIs, CSS, permissions, data, model behavior, or knowledge sources drift.
- Agents can learn to satisfy the grader rather than solve the user problem, so release gates must include human review and production monitoring.
- The business cares about end-to-end success, cost, latency, reliability, reversibility, and user trust, not only response quality.

1.1 Single-turn vs. agent evaluation

HeaderBlue

Single-turn	Prompt + optional expected answer	Response text, meaning match, exact match, general quality	Knowledge Q&A, policy answers, simple RAG
Agent	Task + tools + environment + permissions	Outcome + trace + state change, tool correctness, cost/latency	Workflow automation, email actions, tickets, database updates, multi-step research

2 Define what you are evaluating

Before building test cases, write the agent's mission and boundaries. If the mission is vague, the eval will be vague. If the boundaries are vague, the agent will eventually answer something it should not.

HeaderBlue

Task	What the agent must complete	Answer a policy question; summarize a complaint; draft a case note; create a ticket
Trial	One run of the task	Run the same task multiple times because outputs and tool paths can vary
Trace / transcript	The process record	Input, retrieval, tool calls, intermediate state, output, errors, latency, cost
Grader	How success is judged	Programmatic rule, LLM judge, human review, pairwise comparison
Environment	Execution context	User profile, permissions, tools, connectors, data snapshot, caches, sandbox state
Release gate	What blocks production	Regression score, end-to-end success, cost/latency, anomaly rate, human approval

3 Ten launch-readiness questions

1. Can the team state the agent's core mission in one sentence?
2. Are in-scope and out-of-scope scenarios explicitly documented?
3. Does each key scenario have representative prompts or tasks from real users, internal testing, or known failure modes?
4. Does every task have a success criterion and reference solution or expected evidence?
5. Are positive cases and negative cases both included?
6. For RAG/document AI, does the eval check groundedness, citation behavior, source freshness, and "do not know" behavior?
7. For tool/action agents, does the eval verify the actual state change, not just the agent's final message?
8. Are capability evals separated from regression evals?
9. Are graders calibrated with human review and checked for false pass/fail results?
10. Are release gates and online monitoring rules defined before production access is enabled?

4 Four-stage enterprise evaluation model

HeaderBlue

1. Core	Build a foundational test set for the agent's key scenarios.	Mission, boundaries, scenario list, task prompts, acceptance criteria.
2. Baseline	Run the first eval, record pass/fail, and diagnose failure categories.	Baseline score, failed cases, root-cause analysis, improvement backlog.
3. Expand	Add variation, architecture, edge-case, permission, and tool-use tests.	Capability suite, robustness suite, architecture suite, guardrail suite.
4. Operate	Run evals continuously after changes and in production monitoring.	Release gates, regression suite, scheduled runs, incident-triggered runs, online feedback loop.

5 Build the first 30 tasks

Start with pain, not imagination. The first eval suite should cover real failure patterns and high-impact workflows. A small but sharp set is better than a large synthetic benchmark that nobody maintains.

- Collect real failures from internal testing, customer support, business users, demo reviews, or pilot feedback.
- Rank by user impact, business risk, and frequency.
- For each task, define the required outcome, forbidden behavior, reference solution, and review evidence.
- Promote stable, must-not-break tasks into the regression suite.

6 Evaluation suite design

HeaderBlue

Core regression suite	Does the agent still perform the must-pass scenarios?	Run on every prompt, tool, model, or knowledge change.
Capability suite	Can the agent perform new or hard capabilities?	Run during development and product iteration.
Variation suite	Does success generalize beyond exact wording?	Run before release or after instruction changes.
Architecture suite	Which component failed?	Run after tool, connector, retrieval, or routing changes.
Guardrail / boundary suite	Does the agent refuse, escalate, or downgrade correctly?	Run after policy or safety changes.

7 Grader selection guide

HeaderBlue

Programmatic / rule	Schema, JSON, exact text, tool state, database row, citation presence	Can be brittle if formats vary
LLM judge	Semantic quality, completeness, groundedness, nuanced business rules	Must be calibrated; vulnerable to circularity and prompt injection
Human review	Ambiguous, high-risk, subjective, new failure modes	Slow and expensive; use sampling
Pairwise comparison	Version A vs. version B	Does not always define absolute launch readiness

8 Failure taxonomy

HeaderBlue

Knowledge failure	Retrieved source is missing, stale, irrelevant, or contradicted.	Update source, improve chunking, add freshness checks.
Instruction failure	Agent had the right information but followed unclear or conflicting instructions.	Refine instructions, examples, and boundaries.
Retrieval failure	Correct document exists but was not retrieved or cited.	Tune retrieval, metadata, query expansion, and reranking.
Tool design failure	Tool interface makes mistakes easy.	Redesign tool schema, add examples, require confirmations.
Permission / profile failure	Evaluation profile lacks required access or has too much access.	Create authenticated profiles and permission tiers.
State-change failure	Final answer says done, but ticket/email/calendar/database state is wrong.	Verify actual state after action.

Boundary failure	Agent answers, acts, or reveals information outside allowed scope.	Add action tiers, approval, refusal/escalation rules.
Evaluation failure	Agent was reasonable, but grader or acceptance criteria were wrong.	Adjust test case, calibrate judge, document ambiguity.
Environment failure	API, cache, CSS, data, connector, or sandbox drift caused flake.	Isolate trial state and monitor drift.

9 RAG and document AI checks

- Answer uses the right approved source type and includes citations when auditability matters.
- The agent abstains or escalates when the answer is not present in the approved knowledge base.
- Source freshness is checked; stale documents should not silently override current policy.
- The retrieval trace is reviewable: query, selected chunks, source metadata, and final cited evidence can be inspected.
- For sensitive enterprise contexts, identifiable details are minimized and reviewed before downstream use.

10 Copilot Studio implementation notes

Use built-in evaluation features where they fit, then extend with your own operational checks where needed.

- Create or import a test set from approved questions and expected responses.
- Use single-response tests for isolated question-answer behavior.
- Use conversation tests where context retention, escalation, or multi-turn flow matters.
- Select multiple methods when needed: general quality, compare meaning, classification/custom business rules, and tool/capability checks.
- Assign the right user profile so the evaluation exercises real permissions and connections.
- Review per-question details, knowledge sources, transcripts, and tool/activity information; do not rely only on aggregate score.

11 Release gates

HeaderBlue

Regression gate	Must-pass suite is below threshold.	Block release and inspect recent changes.
End-to-end success gate	Workflow success rate is below business threshold.	Keep read-only or draft-only.
Cost/latency gate	Token cost, tool calls, retries, or latency exceed budget.	Downgrade mode, simplify path, or require approval.
Anomaly gate	Failure, refusal, retry, tool error, or drift spikes.	Circuit breaker: pause, route to human review, or revert.

For write actions and irreversible tools

Use action tiers, circuit breakers, and human approval. A production agent that can change records, send messages, trigger transactions, or update workflows needs stronger gates than a read-only Q&A agent.

12 Online monitoring loop

- Monitor success rate, retry rate, tool error rate, latency, cost, human handoff rate, and failure category distribution.
- Sample production traces every week and compare them with automated grader results.
- Convert top failures into new evaluation tasks and add them to the appropriate suite.
- Track model, prompt, tool, data, and knowledge-source versions for every evaluation run.
- Report quality in business language: which scenario failed, why it failed, and what fix is planned.

13 Healthcare, HR, and enterprise data-product lens

HeaderBlue

Healthcare quality / complaints	Theme extraction, contributing factors, escalation signals, residual PII, patient-safety relevance.	Human review required for sensitive narratives and quality-improvement interpretation.
HR / employee service	Policy accuracy, personalization, employee tenure, permissions, escalation to HR.	Never infer private employee data without authorized context.
Finance / operational analytics	Definitions, metrics, risk, approvals, audit trail, cost and latency.	Every automated action needs a reversible path or approval point.
Data products	Decision cadence, data freshness, metric ownership, adoption signal.	The eval should test whether the system supports a real decision, not only whether it answers a question.

14 Copy-paste launch checklist

Mission and boundaries documented.

First 30 tasks selected from real failures or high-impact scenarios.

Every task has acceptance criteria and a reference solution or evidence target.

Harness runs with isolated state, controlled permissions, and clean caches.

Trace captures input, retrieval, tool calls, intermediate state, output, cost, latency, and errors.

Programmatic, LLM-judge, and human-review graders are assigned by dimension.

Capability suite and regression suite are separated.

Release gates are defined for regression, end-to-end success, cost/latency, and anomaly spikes.

Online monitoring loop turns weekly top failures into new eval tasks.

References and further reading

- Microsoft: Review the agent evaluation checklist: https://learn.microsoft.com/en-us/agents/agent-evaluation/evaluation-checklist
- Microsoft Copilot Studio: About agent evaluation: https://learn.microsoft.com/en-us/microsoft-copilot-analytics-agent-evaluation-intro
- Microsoft Copilot Studio: Choose evaluation methods: https://learn.microsoft.com/en-us/microsoft-copil analytics-agent-evaluation-overview
- Microsoft Copilot Studio: Create a single response test set: https://learn.microsoft.com/en-us/microsoft-copilot-studio/analytics-agent-evaluation-create
- LangChain: Agent Evaluation Readiness Checklist: https://www.langchain.com/blog/agent-evaluation-read
- Anthropic: Demystifying evals for AI agents: https://www.anthropic.com/engineering/demystifying-evals-
- Ribeiro et al.: Beyond Accuracy: Behavioral Testing of NLP Models with CheckList: https://arxiv.org/abs/2005.04118

Attribution note

This checklist is an original synthesis and packaging for enterprise AI evaluation practice. It is inspired by public evaluation concepts from Microsoft, LangChain, Anthropic, and the broader NLP evaluation literature, but it is not a copy of those materials.