



Agentic AI Foundation

AGNTCon + MCPCon North America

Where the **agentic stack** is being built.

Stop Writing Agents, Declare Them

Declarative Agent Architecture in Production
Chris Knuteson · A Place for Mom

“Zero failed.” The PR said v47 → v41

A routine prompt-release PR for our production chat agent

17 promoted · 0 skipped · 0 failed

version: 47 → version: 41

“Is it expected that the version number is going down?”

Metadata-only diff, closed unmerged, version guards added two weeks later. Nobody could say from the PR which agent production was actually running.

Agents for families choosing care

A Place for Mom

The nation's leading senior care advisor platform · 15,000+ communities and home care providers in our network

Grace

A Place for Mom's digital advisor, drawing on 25 years of advisor expertise, resources and family insights to help families understand and explore senior care options.

2,526

agent runs

chat-agent runs · 1,767 sessions ·
404 users · 30 days

19,362

model calls

generations, incl. speech and
embeddings

17

prompt packs

on the production label, resolved at
runtime

Two systems in this talk. Grace runs in PROD. The PR reviewer you will see is hosted in DEV and reviews the production PRs we merge; PROD is its promotion target.

The app built the agent at runtime

```
const label = options?.label ?? defaultLabel; // "production"
```

```
const cacheTtlSeconds = options?.cacheTtlSeconds ?? 60;
```

```
langfusePrompt = await langfuseClient.prompt.get(
```

```
  packId, { label, cacheTtlSeconds },
```

```
);
```

Grace · Langfuse adapter, label path, identifiers trimmed. An immutable-version path exists beside it.

chat-persona-base, today

43 Langfuse versions · its own version field says **48** · two counters, one agent — nothing that names the whole of it

What a turn depended on

Git

the app commit

Langfuse

the production label, re-read every 60 s

Bundle

base + profile packs merged at run time

Declare the agent, so the run has a receipt

AgentSpec one versioned document that declares one agent: identity, instructions, skills, tools and connections, runtime, budget, policy.

```
"id": "review.apfm-agentic-review",
"runtime": "bedrock-agentcore-harness",
"model": "anthropic/claude-opus-4-7",
"harness": { "max_iterations": 80, "max_tokens": 16000 },
"spec": {
  "instructions": "You are the APFM Agentic PR Reviewer ...",
  "skills": [ "apfm-review-orchestration", ... 16 refs ],
  "tools": [ "review.call-all-dimensions",
    "connection.github-review-readonly" ],
  "surface": "annotate"
}
```

Runtime

AWS Bedrock
AgentCore: a managed
agent loop, one isolated
session per run

Carries

instructions and 16 skill
references

Requests

capabilities by name;
grants stay with the
platform

Actual reviewer source, selected fields. The compiler turns instructions into the harness system prompt.

One manifest shows the blast radius

```
"root_agent": "review.apfm-agentic-review",
```

```
"agents": [
```

```
  { "id": "review.child.bug-integration",
```

```
    "path": "child-bug-integration/agent.json" },
```

```
  { "id": "review.child.perf-userpov", ... },
```

```
  { "id": "review.child.alignment-comments", ... } 
```

```
],
```

```
"skills": [ ... 16 registered skill entries ... ]
```

Root

the entry point and its handoff tool

Children

three specialists, each its own AgentSpec

Skills

versioned bodies the closure depends on

A behaviour change to one specialist is a diff against this manifest. Reviewers see what else moves before it ships.

Plain JSON. What makes it a desired state is the contract the compiler enforces, not the file format.

Validate, submit, then activate

Pull request compile and dependency-check the affected definitions. Never publishes.

Merge submit the changed source as a DEV candidate, named by its bundle digest.

Promotion stage · run the canary suite sealed into the bundle · activate · roll back.
Explicit steps, each with a receipt.

One push, one agent, one chain

commit 2687ce60 → bundle **614017b6...** → agent **incident-triage-live.release-614017b6...** →
canary run succeeded 11 min later

The release name carries the source digest, so the receipt answers “what ran?” by itself. Live for incident-triage, not yet for the reviewer.

Rollback targets age out

Attempted

roll the incident-triage agent back from v5 to v4 in DEV

Refused

blocked_rollback_runtime_unavailable — v4 was sealed on a platform release the next Gateway deploy retired

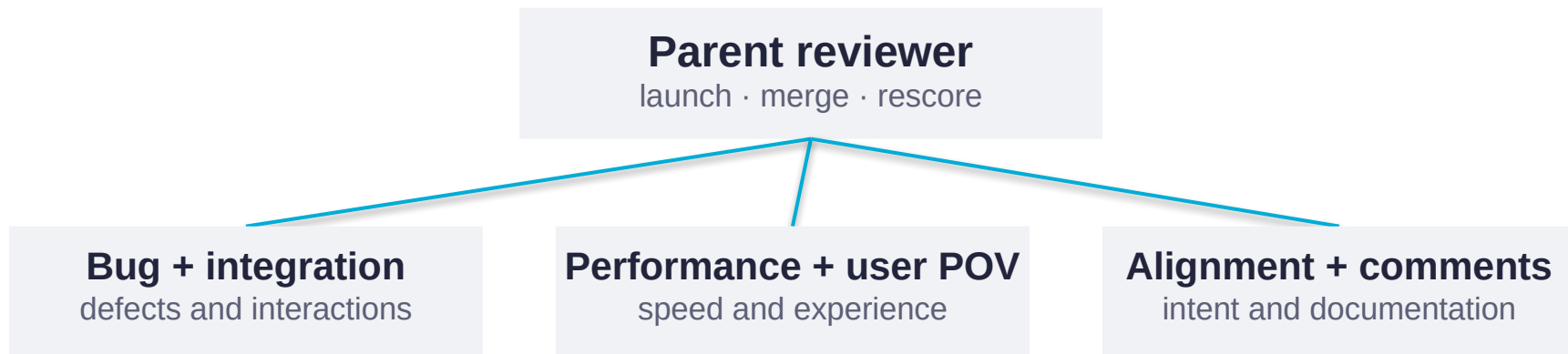
Fix

seal a fresh pair (v6, v7) on the current platform release, then prove v7 → v6

Tradeoff we did not expect: an agent release is pinned to the platform release it was sealed on. Rollback targets age out when the platform moves.

Exercised on our incident-triage agent, DEV. The reviewer still lives in the GUI and is synced back to Git — slide 12.

One parent, three specialists



AgentCore runs the fan-out outside Grace's application runtime. Each box is its own managed harness with its own session, budget and receipt.

A boundary chosen before child runs, sessions and lifecycle became Grace's operating overhead — not because Grace is saturated.

Real PRs across the organisation

26

repos

of 29 subscribed produced
a review

1,928

reviews delivered

of 3,850 workflow runs, 30
days

1,017

pull requests

distinct PR branches
reviewed

7.4

median review, min

p90 11.1 · 84 runs, one
repo

Behind them

1,283 parent + 4,099 specialist runs in 14 days of control-plane receipts · 14 failed invocations of 5,396 · 143 retries

Window

last 30 complete days of GitHub run records; the Gateway reviewer served every sampled run after the cutover

Not shown

finding accuracy, acceptance, token cost — a run record measures none of these (definitions: appendix)

Completion $\neq$ accuracy

```
coverage.get("complete") is True  
and set(expected) <= set(succeeded)
```

```
and _attempts_resolved(coverage)
```

```
and coverage.get(  
    "narrowed_to_single_agent"
```

```
) is False
```

*Trusted delivery gate · the control plane builds the coverage receipt
from its own child-run evidence, never from the model*

Proves

the configured panel ran and
the result reached GitHub

**Does not
prove**

findings are correct, useful or
accepted — no
precision/recall measured

Still off

approval and auto-merge: the
gate exists, the kill switch
stays on

A parent saying “Clean” is not evidence its specialists ran. The receipt is.

Same drift, new platform

Last week · a PR titled “sync the review agents with what is live in DEV”

Week 0 the review agents are exported from the DEV GUI into the repo

Week 3 review-quality prompts change live in the GUI

Week 5 a child moves models; fallback chains are added live

Week 5 the sync PR copies live back into Git. Merging deploys nothing.

An export is a snapshot. Custody has to be enforced — the reviewer has the runtime story, not yet the custody story.

What is pinned, what still moves

```
request["qualifier"] = credential.qualifier # or DEFAULT
```

```
request["model"] = {"liteLlmModelConfig": {
```

```
    "modelId": f"openai/{credential.revision_spec['model']}",
```

```
    "apiBase": _dispatch_gateway_api_base(), ... }}
```

Our invoke path, trimmed. AgentCore would accept a system prompt, tools, skills and budget per call; the trusted worker never sends them, and callers never touch this request.

Retained configuration identifies the definition that ran. It freezes neither behaviour nor the systems the agent talks to.

Pinned

instructions → system prompt;
skills from sealed evidence;
the model named in the
revision

Follows

the reviewer runs on
DEFAULT, which tracks the
latest of its 30 runtime
versions

Moves

model outputs, tool-backed
systems, external inputs;
legacy evidence-less revisions
can load mutable skills

Four layers: what would port

Layer	What is in it	Would it port?
AgentSpec + release identity	ids, instructions, skill refs, manifest, bundle digest	Yes. Plain documents and digests.
APFM policy and promotion	compile, authorization, candidate → canary → activate → rollback	The mechanism, not the code: it is written against the Gateway.
AgentCore execution	harness runtimes, versions and endpoints, microVM sessions, InvokeHarness	No. An adapter re-implements compile, skill mounting, session and child-run evidence, credentials.
Consumers and delivery	GitHub router, coverage gate, posting, approval policy	Yes, by construction: it only sees receipts.

JSON is not portability. Still coupled: harness semantics (iterations, stop reasons), the child-handoff protocol, log and metric shapes.

Build versus buy

Bought

AgentCore for execution (harness loop, microVM sessions) · Langfuse for authoring, tracing, evals · GitHub Actions for delivery

Built

the contract, the compiler, the candidate → canary → activate → rollback path, the coverage receipt. No vendor names the closure as one release across all four layers.

AgentCore?

it versions a runtime's container config and DEFAULT follows latest. Our question was the definition — instructions, skills, children, policy — with promotion evidence.

Cost

compiler maintenance, contract evolution, running the promotion path: a platform team's roadmap, not a side project

Would buy

a registry or linter the day a standard covers the contract. AgentSpec borrows Kubernetes-style spec/status shape on purpose so one can.

We buy execution and observability. We build the thing that answers “what ran?”

What the abstract promised

One spec, one job

Shipped. Four AgentSpecs in one manifest: the reviewer parent and three specialists.

Policy on the key

Shipped in the Gateway. Every model call passes auth, model allowlist, budget, rate limit, PII scrub, kill switch and audit hops; the reviewer's key is minted per dispatch and revoked after.

Prompts in a registry

Partly. The reviewer carries inline instructions plus 16 versioned skills; registry references resolved into a release are next.

Tools are refs

Shipped. Tools and connections are named in the spec; grants stay with the platform.

Zero-change swaps

Honest: model and runtime are fields in the spec. The contract and the receipts port; the harness adapter does not.

Web, A2A, MCP

Not built. The reviewer is invoked over a signed GitHub channel; parent → child handoffs are the only agent-to-agent path today.

Three shipped, two partly, one ahead. That is the honest account the abstract promised.

Where this goes next

Now

incident-triage on Git custody with sealed canaries and a rollback proof; the reviewer serving 26 repos from DEV, GUI-managed

Next

hand the four review agents to Git custody; run the reviewer on a release-named endpoint instead of DEFAULT

Then

lint declarations against the same compiler; resolve prompt-registry references into the release

Proposal

promote the reviewer to PROD through the same candidate → canary → activate path. Not started.

Sequenced by blast radius: prove the path on an ops agent, then move the one everyone depends on. “Next” onward are proposals; the platform costs are real.

Lessons for another platform team

- 1 Name the release in the thing that runs**
a digest in the agent name turns a log line into an identity receipt
- 2 Separate submit, activate and roll back**
and expect rollback targets to age with the platform release
- 3 Get coverage from the control plane**
a model saying “done” is not evidence; child-run records are
- 4 Keep write authority out of the agent**
the runtime returns text; trusted code posts, approves, merges
- 5 Treat exports as snapshots**
custody is a handoff you enforce, or the GUI quietly wins

Everything above came from running agents and getting the release question wrong first.

Ship the agent. Keep the receipt.

Name the release before you trust the run.

Chris Knuteson · Director of Agentic, A Place for Mom

Questions

Session feedback QR
replace before export

Scan to leave session
feedback

Appendix · Grace vs reviewer, same definitions

	Grace (chat, PROD)	Gateway reviewer (DEV)
Definitions	17 prompt packs on the production label; the agent is assembled in-process	4 AgentSpecs + 16 skills in one manifest, compiled to harness config
Usage (30 d)	2,526 agent runs · 11,122 traces · 19,362 model calls	1,928 reviews delivered · 1,283 parent + 4,099 child runs (14 d receipts)
Identity	app commit + label at run time; versions flattened into a string	sealed revision + session id + coverage receipt
Custody	Langfuse label, re-read every 60 s	GUI-managed, synced to Git last week; Git binding next
Execution	one NestJS process, OpenAI Agents SDK	AgentCore harness sessions, parent + 3 children

Different workloads: a chat turn and a ten-minute review. Latency and cost are not compared on purpose.

Appendix · Versioning fixes in Grace

The PR

promotion PR proposed metadata v47 → v41; closed unmerged

+2 weeks

semantic pack versions separated from Langfuse counters; downgrades and missing bumps rejected

+4 weeks

repo v1 → production v5 bookkeeping; content-identical, no behaviour change

Today

chat-persona-base: 43 Langfuse versions, pack version field 48 — the counters still differ

Pins and version guards fixed prompt movement. They never named the whole agent: instructions, skills, children, runtime, policy.

Appendix · Metric definitions

Grace usage	Langfuse, environment prod, last 30 complete days; agent run = root trace “Agent workflow”; model call = GENERATION observation
Reviews delivered	GitHub Actions, workflow “Agentic PR Review”, 29 subscribed repos, same 30 days; success = run conclusion success
Pull requests	distinct (repo, head branch) with a successful run — an upper bound on PRs
Parent / child runs	DEV control-plane log lines “channel invoke run ... succeeded”, last 14 days (log retention)
Latency	“Run review agent in Grace” step duration, grace-ai-gateway only, 84 completed runs in the last two weeks
Not measured	finding precision/recall, acceptance, token cost, PROD traffic, per-run engine for all 3,850 runs

Read-only via platform-mcp (GitHub, AWS) and the Langfuse MCP. Exact windows, dates and sources in research.md.

Appendix · Three receipts, one review

GitHub run agent step 5 min 5 s · review posted 4 s after the step ended · HTTP 200

Control plane parent succeeded 2 s before the step ended · coverage v2: 3 expected, 3 succeeded, 0 failed, 0 recovered

Policy artifact result SHA-256 bound to repo, run and attempt · risk 4/10, floor 3/10 · approve=false

What this run does not prove: which source commit produced the reviewer that ran. That join exists for the Git-managed agent, not yet for this one.

One grace-ai-gateway PR; the run whose policy artifact was preserved locally. Identifiers in research.md.