

— THE COMPLETE GUIDE

The Runbook to Skill Handbook.

How to take the method that lives in your best engineer's head, write it down as a governed skill, and let the whole team — and every agent — run it the same way, every time.

12 chapters 3 frameworks 5-part recipe

— THREE FRAMEWORKS YOU CAN REUSE

01

The Method Audit

Six questions that pick the runbooks worth turning into skills.

02

The Autonomy Classes

discover, manage, automate — and what earns each step.

03

The Skill Readiness Review

Ten honest checks before a skill runs unattended.

— WHO THIS IS FOR

- SRE and platform leads tired of being the escalation path
- NOC and DBA teams who want juniors running senior methods
- Engineering managers watching expertise walk out the door

— WHAT A WRITTEN-DOWN METHOD IS WORTH — PUBLISHED BENCHMARK RANGES

40–60%

LOWER MTTR

~90%

LESS ALERT NOISE

1/3 week

BACK FROM FIREFIGHTING

Day 1

NEW-HIRE RAMP

A junior directing a skill follows the same steps as your best senior.

Versioned, routed, governed skills — one engineer's best method, runnable by the whole team.

— CONTENTS

What this handbook covers

This is for the people who get the 3am phone call because they're the only one who knows the trick. It assumes your team already has good runbooks, or at least good instincts. The question it answers is how to make those instincts run without you.

01 Why expertise does not scale on its own The method was never the bottleneck	p03	02 A recipe, not a prompt Why structure beats clever wording	p04
03 Framework 1 — the Method Audit Which runbooks are worth converting	p05	04 From SOP to skill A worked example, line by line	p06
05 Routing: the right method first Why the search phase costs so much	p07	06 Constraints that travel Guardrails that ship inside the work	p08
07 Framework 2 — the Autonomy Classes discover, manage, automate	p09	08 Output schemas and composition Skills that feed other skills	p10
09 Versioning and calibration Noticing drift before it hurts	p11	10 Build, clone or install Catalog, workspace, marketplace	p12
11 Framework 3 — the Skill Readiness Review Ten questions before promotion	p13	12 Getting started Your first skill this week	p14

— THREE FRAMEWORKS YOU CAN REUSE

The Method Audit

Six questions that pick the runbooks worth turning into skills.

The Autonomy Classes

discover, manage, automate — and what earns each step.

The Skill Readiness Review

Ten honest checks before a skill runs unattended.

Chapters 1–2 make the case for structure over prompting. Chapters 3–6 cover writing and routing skills safely. Chapters 7–10 cover autonomy, composition and keeping skills accurate over time. Chapters 11–12 get you to a first promoted skill this week.

— BRING THESE TO THE FIRST READ-THROUGH

- A month of recent tickets or pages, exported
- The runbooks or SOPs your team actually uses
- Twenty minutes with the person everyone escalates to
- A list of the systems each fix touches

You don't need a transformation programme. You need one boring method written down properly.

— CHAPTER ONE

Why expertise does not scale on its own

Every operations team has a person who is, quietly, the runbook. They are brilliant, they are exhausted, and they are one resignation letter away from taking half your incident response with them.

The problem isn't that the knowledge is missing. It's that it lives in a format only one human can execute. The wiki page exists, but it's three versions behind and skips the step everyone 'just knows'. So the on-call engineer reads it, gets stuck on step four, and phones the person who wrote it. Multiply that by every recurring incident type and you have a team whose capacity is capped by one calendar.

A runbook nobody trusts is just a longer way of saying "call Dave".

— FOUR SIGNS YOUR EXPERTISE IS STUCK IN ONE HEAD

- The same person gets paged for the same class of incident every month.
- New hires take a quarter or more before they can handle on-call alone.
- Your runbooks are accurate — as of two reorgs ago.
- When the expert is on holiday, MTTR visibly gets worse.

— WHAT THAT COSTS, IN PUBLISHED BENCHMARKS

3-6 mo

TYPICAL NEW-HIRE RAMP

1½ week

LOST TO FIREFIGHTING

\$300K+

COST OF ONE OUTAGE HOUR

Chapter 2 looks at why simply prompting a model with the runbook doesn't fix this — and what does.

— LISTEN TO THE ESCALATION CALLS

WHERE ESCALATIONS COME FROM	WHAT IS ACTUALLY MISSING
"Step four doesn't work any more"	A runbook that has drifted from the estate
"Which dashboard do I even open?"	The declared list of systems the fix reads
"Am I allowed to restart this?"	A written limit, instead of a judgment call at 3am
"What did you find last time?"	A consistent output from previous investigations

— CHAPTER TWO

A recipe, not a prompt

The tempting fix is to paste the runbook into a model and ask it to follow along. It works in the demo. It drifts in production, because a prompt is a suggestion, not a contract.

— THE DIFFERENCE, SIDE BY SIDE

	A PROMPT	A SKILL
Order of steps	Whatever the model decides this time	Fixed, written down once
Systems touched	Anything its credentials reach	Only the declared tool calls
Limits	Hoped for, in the wording	Enforced constraints, every run
Output	Prose, different each time	A schema other skills can consume
Dependencies	Discovered when they fail	Declared connectors, checked first

— THE FIVE PARTS OF A RECIPE

01

Steps

The order an expert would work in, written down once.

02

Tool calls

Exactly which systems it may query, with which parameters.

03

Constraints

What it must never do. The guardrail ships inside the skill.

04

Output schema

A predictable shape other skills and workflows can consume.

05

Connectors

The integrations it depends on, checked before routing.

If you can't predict the shape of what comes back, you can't build anything on top of it.

— WHY THE DEMO AND 3AM BEHAVE DIFFERENTLY

WHAT DRIFTS IN A PROMPT

- Step order changes between runs
- New tools creep in when credentials allow
- Limits get paraphrased away
- Output shape changes with the phrasing

WHAT HOLDS IN A SKILL

- Steps fixed and versioned
- Only the declared tool calls
- Constraints checked every run
- One schema, every time

— CHAPTER THREE · FRAMEWORK ONE

The Method Audit

Not every runbook deserves to be a skill. Before you write one, run your team's recurring work through six questions.

QUESTION	A STRONG ANSWER
How often does this come up?	Weekly or more across the team.
Is there one agreed method?	Yes — or two, clearly split by a known condition.
Which systems does it read?	A short, nameable list you can connect.
What must it never do?	You can say it in one line — that is your first constraint.
What does "done" look like?	A result with a clear shape: cause, evidence, next steps.
Who currently gets escalated?	A specific person — whose afternoon you are about to give back.

Running it

Take a month of tickets and pages. Group them by the method that fixed them, not by the alert that fired — different alerts often share one fix. Score each group against the six questions. The winners are rarely the dramatic incidents; they're the boring, frequent ones the same person has solved forty times.

— SCORING IT

1

Five or six yeses

A strong first skill. Start it in discover.

2

Three or four

Worth writing, but pin down the method first.

3

Two or fewer

Leave it. The method is not settled yet.

The best first skill is the one your expert is most bored of explaining.

— A WORKED EXAMPLE

CANDIDATE	FREQ.	ONE METHOD?	LIMITS CLEAR?	SCORE
Crash-loop diagnosis	Daily	Yes	Yes	6 / 6
Pool saturation	Weekly	Yes	Yes	5 / 6
Certificate expiry checks	Weekly	Yes	Mostly	5 / 6
Novel multi-region outage	Rare	No	No	1 / 6

— CHAPTER FOUR

From SOP to skill

Writing a skill is mostly translation. The SOP already has the steps; your job is to make the implicit parts explicit — the systems, the limits, and what the answer should look like.

before

the SOP

```
# Connection pool SOP (wiki, 2024)
1. Check active sessions in the DB
2. Look at Datadog latency for the service
3. See if pool maxed out around the spike
4. Find the query holding locks
5. DON'T kill anything without the DBA
6. Post findings in #db-oncall
```

after

the skill

```
steps:
  - pull active sessions + wait events
  - correlate pool saturation vs latency
  - rank blocking queries by impact
constraints:
  - read_only: true
output: root_cause, evidence[],
       confidence, next_actions[]
```

— WHAT THE TRANSLATION ACTUALLY CHANGES

IN THE SOP	BECOMES IN THE SKILL
"Check the DB" and "look at Datadog"	Declared connectors: postgres, datadog
"See if the pool maxed out"	A step with a defined correlation, not a vibe
"DON'T kill anything"	A constraint: read_only, never terminate a session
"Post findings"	An output schema anyone — or any workflow — can read

The shouting in step five was always the most important line in the SOP. In the skill, it finally gets enforced.

— THE TRANSLATION CHECKLIST

1

Name the systems

Every "check X" becomes a declared connector.

2

Name the limits

Every "don't" becomes a constraint.

3

Name the result

Every "post findings" becomes a schema.

If a step says "use your judgement", the skill is not finished yet — the judgement is the part to write down.

— CHAPTER FIVE

Routing: the right method first

Most of the time lost in an incident isn't spent fixing it. It's spent trying the wrong approach first, then the second-wrong one. Routing exists to skip that search phase.

— FOUR CHECKS, EVERY CANDIDATE, EVERY QUERY

Relevance**embedding**

Does the question actually match what this skill says it is for?

Availability**connector**

Is the system this skill depends on connected right now?

Skill state**state**

Enabled, drafted or deprecated — only live skills are routed.

Permission**RBAC**

Your role decides what runs. Denied skills are dropped, not hidden.

32

Candidate skills

27

Pass persona & state

3

Relevant and permitted

✓

Tools invoked

1st try

the right method runs first,
not fourth

Notice what routing is not: it is not a similarity search that grabs the nearest match. A skill that matches the question perfectly but depends on a disconnected system is worse than useless — it fails halfway through. Checking the connector first means the router picks the next best skill that can actually finish.

Denied skills are dropped, not hidden. You should always be able to see why something did not run.

— WHERE THE SEARCH PHASE GOES

WITHOUT ROUTING	WITH ROUTING
Try the obvious skill first	Relevance ranks every candidate
Discover the connector is down halfway	Availability checked before anything runs
Run a deprecated version by accident	Only enabled versions are ever routed
Hit a permission error mid-incident	Denied skills dropped up front, visibly

● Routing is testable before production.

● The router explains every skip, not just every pick.

— CHAPTER SIX

Constraints that travel

The usual place for guardrails is the agent. The problem is that agents multiply — and every new one is a fresh chance to forget a rule. Put the rule in the skill instead.

— A CONSTRAINT BLOCK

```
constraints:
- read_only: true
- max_rows: 5000
- never: [terminate_session, drop_index, restart_primary]
- targets: tagged env:staging only
- requires_approval_on: [prod-tier-1]
```

— WHAT EACH KIND PROTECTS AGAINST

CONSTRAINT TYPE	WHAT IT PREVENTS
read_only	Any write, however well-intentioned
never: [...]	Specific destructive actions, by name
max_rows / limits	An investigation turning into a load test
targets	Acting outside the tagged blast radius
requires_approval_on	Unattended change on the systems that matter most

A new agent inherits every constraint of every skill it calls. Nobody has to remember anything.

— WHERE THE RULE SHOULD LIVE

GUARDRAILS ON THE AGENT

- Rewritten for every new agent
- Easy to forget on the fifth one
- Invisible to the skill author
- Reviewed per agent, slowly

GUARDRAILS IN THE SKILL

- Written once, inherited everywhere
- Impossible to forget by construction
- Owned by the person who knows the risk
- Reviewed once, per version

— CHAPTER SEVEN · FRAMEWORK TWO

The Autonomy Classes

Three classes, declared on the skill itself. Each step up is earned by a track record, not granted because the skill looks good.

- 1 discover.** Investigates and reports. Cannot change a thing. Safe on any site.
Earned by: nothing — every new skill starts here
- 2 manage.** Changes tagged targets within a blast radius, with approval where the tier requires it.
Earned by: a clean discover record and clearly tagged targets
- 3 automate.** Runs unattended when its trigger fires.
Earned by: a sustained acceptance rate on real incidents

Why the class lives on the skill

If autonomy were an agent setting, one permissive agent could run a skill far beyond what its author intended. When the class is part of the skill, the ceiling is fixed at the source: an automate-class agent calling a discover-class skill still only discovers.

— BEFORE MOVING A SKILL UP A CLASS

- Has it run in discover long enough to show a stable acceptance rate?
- Are its manage targets tagged, not inferred?
- Would you accept its output without re-checking — every time?
- Is there a named owner who reviews its runs?

— WHAT A PROMOTION LOOKS LIKE ON PAPER

```
promotion record stale-snapshot-cleanup
discover → manage after 6 weeks · 212 runs · 97% accepted
manage → automate after 4 months · 1,840 runs · 99% accepted
reviewed by: platform-lead rollback: v5 retained
```

— WHAT EACH CLASS MAY ACTUALLY TOUCH

CLASS	MAY READ	MAY CHANGE	NEEDS A PERSON
discover	Its declared connectors	Nothing	Never
manage	Its declared connectors	Tagged targets in its radius	On the tiers you name
automate	Its declared connectors	Tagged targets in its radius	Only when a check fails

— CHAPTER EIGHT

Output schemas and composition

A skill that returns a paragraph is a dead end. A skill that returns a predictable shape becomes a building block — for other skills, for workflows, for agents.

— THE SAME SHAPE, EVERY RUN

```
output:
  root_cause: "pool saturation on payments-db"
  evidence: [wait_events, latency_p99, pool_usage]
  confidence: 0.87
  next_actions: ["raise pool ceiling", "index orders.created_at"]
```

— SKILLS FEEDING SKILLS

01

deploy-timeline

what changed, when

02

latency-correlation

where it slowed

03

db-pool-diagnostics

why it slowed

04

workflow

decide and act

Because each output has a known shape, the next step doesn't need to interpret prose. A workflow can branch on confidence, an agent can pick next_actions, and a report can be built without anyone copying text out of a chat window.

Composition is where skills stop being a convenience and start being infrastructure.

— ONE SHAPE, MANY READERS

WHO READS THE OUTPUT	WHAT THEY USE
The next skill in the chain	root_cause and evidence, to narrow its own search
A workflow	confidence, to branch between remediate and escalate
An agent	next_actions, to propose what to do
An auditor	The whole record, unchanged, after the fact

— THREE FIELDS THAT MAKE OUTPUT USEFUL

root_cause

One sentence a human can check.

evidence[]

The signals that support it, linked.

confidence

A number the next step can branch on.

— CHAPTER NINE

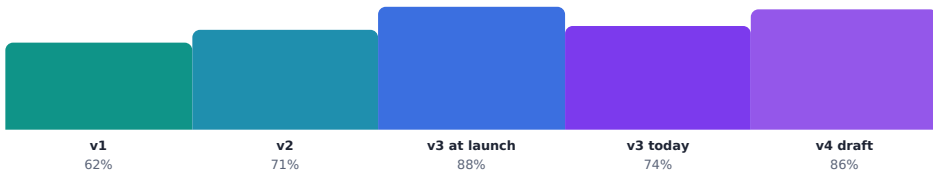
Versioning and calibration

Your estate changes every week. A skill written against last year's instance types slowly gets worse, and nobody notices until it confidently gets something wrong.

— DRIFT, CAUGHT BY THE NUMBERS

```
aws-investigation
v3 enabled calibration 74% # drifted 14 pts in 30 days
v4 drafted calibration 86% on replay
recommendation: promote v4 · keep v3 for rollback
```

— CALIBRATION ACROSS VERSIONS



Illustrative version history. The drop from 88% to 74% is what a changing estate looks like when nobody is measuring.

- Every version keeps its own calibration score.
- Drafts are replayed against real incidents before promotion.
- Rollback is one step, because the previous version is never deleted.
- Drift triggers a recommendation, not a surprise.

— DRIFT HAS A SIGNATURE

SIGNAL	WHAT IT USUALLY MEANS
Acceptance falling on one incident type	A new pattern the skill was never written for
Calibration drifting slowly	The estate changed — new SKUs, versions, regions
A connector routed around often	An integration quietly degrading
Rising overrides by engineers	The method is right but a step is out of date

— CHAPTER TEN

Build, clone or install

Writing from scratch is the slowest option and usually the wrong one. Most of the time, someone has already solved eighty percent of your problem.

Company catalog**Built in by UnityOne**

32 production-tested skills, versioned and maintained. Start here.

Workspace**Yours and your team's**

Clone a built-in skill and tune it, or write one from your SOP.

Marketplace**Installed on demand**

Specialist skills, recommended when Lumi spots a gap.

— CHOOSING WHERE TO START

SITUATION	BEST MOVE
A built-in skill fits your estate as-is	Use it — enable it for the right persona
It fits, but your method differs slightly	Clone it and tune steps and constraints
Lumi flags a recurring gap	Install the recommended marketplace skill
Your method is genuinely unique to you	Write it from your SOP in the workspace

Cloning keeps the original intact. You get your version, and you still benefit when the built-in one improves.

— WHEN LUMI RECOMMENDS INSTEAD OF YOU GUESSING

Add Memory Leak Detection

Pattern fired 14x last week; closest skill managed 76% acceptance. Expected gain ~12 points.

Refresh AWS few-shot examples

Calibration drifted 14 points; newer instance families missing. Drafted v4 expected 74% → 86%.

— A RULE OF THUMB

- Try the built-in skill as-is before cloning anything.
- Clone when your method differs in a step or two.
- Install when Lumi shows a specific, recurring gap.
- Write from scratch only when the method is truly yours.

The fastest skill to write is the one somebody already maintains for you.

— CHAPTER ELEVEN · FRAMEWORK THREE

The Skill Readiness Review

Ten questions before a skill runs without someone watching. You will not pass all ten on day one — the point is knowing which ones you haven't earned yet.

01

Does it follow one agreed method, written as ordered steps?

02

Are its tool calls the complete list — nothing implied?

03

Is at least one hard constraint declared?

04

Does its output match a schema other steps can consume?

05

Are its connectors declared and currently live?

06

Is its autonomy class set, not defaulted?

07

Has it been replayed against real past incidents?

08

Is its calibration tracked on the current version?

09

Can you roll back to the previous version in one step?

10

Is there a named owner who reviews its runs?

— SCORING IT

1**Eight or more**

Ready to move up a class.

2**Five to seven**

Keep it in discover and close the gaps.

3**Four or fewer**

Still a draft. That is fine.

— WHERE MOST SKILLS FALL SHORT

COMMON GAP	USUAL FIX
No hard constraint declared	Write the one thing it must never do
Output is prose	Define root_cause, evidence, confidence, next_actions
Never replayed on real incidents	Replay last month before promoting
No owner	Name the person who reviews its runs

A skill that passes six of ten is not a failure. It is a skill that knows exactly what it still has to prove.

— CHAPTER TWELVE

Getting started

None of this needs a programme. It needs one boring, recurring method, a clone of the nearest built-in skill, and a week of it running in discover.

STEP	WHAT TO DO	WHAT YOU GET
1 · Run the Method Audit	A month of tickets, grouped by the fix	A shortlist of real candidates
2 · Pick the most boring one	Frequent, well understood, low drama	A first skill you can get right
3 · Clone the closest built-in	Start from the catalog, not a blank file	Most of the work already done
4 · Write the constraints first	What it must never do, in one line each	A skill risk can approve
5 · Test the router	Check which queries select it, and why	No surprises in production
6 · Run in discover, then review	Replay, measure acceptance, promote on evidence	A skill the team trusts

— YOUR FIRST WEEK, ROUGHLY

1 Monday Run the audit on last month.	2 Tuesday Clone the closest built-in skill.	3 Wednesday Write constraints; test the router.	4 Thu-Fri Run in discover; review every result.
---	---	---	---

— FIVE THINGS WORTH TAKING AWAY

- The method was never the bottleneck — its format was.
- Constraints belong in the skill, not the agent.
- A measured skill stays accurate; an unmeasured one drifts.
- Structure beats clever prompting, every time.
- Autonomy is earned by acceptance rate.

Ready to write your first skill?

Open the catalog, clone the closest match, and run it in discover this week.

Pick the dull one. Dull skills are the ones that quietly give your team its evenings back.

— IN CLOSING

Write it down once. Let everyone run it.

Your best engineer's method is already the most valuable thing your team owns. Skills just change its format — from a habit that needs them in the room to a recipe anyone can direct, with the limits written in.

40–60%

LOWER MTTR

~90%

LESS ALERT NOISE

1/3 week

BACK FROM FIREFIGHTING

Day 1

NEW-HIRE RAMP

— THE THREE FRAMEWORKS, IN ONE PLACE

The Method Audit

Six questions that pick the runbooks worth turning into skills.

The Autonomy Classes

discover, manage, automate — and what earns each step.

The Skill Readiness Review

Ten honest checks before a skill runs unattended.

Start with the catalog. Clone one skill. Run it in discover for a week. Then look at the acceptance rate and decide — on evidence — what it has earned.

— THE WHOLE HANDBOOK, IN THREE WORDS

1**Clone**

Start from the catalog.

2**Constrain**

Write the limits first.

3**Prove**

Run in discover, then promote on evidence.

The goal was never automation for its own sake. It was making sure the right method runs, whoever happens to be on call.

Turn your best runbook into a skill.

Start from a built-in skill, clone it, and make it match how your team actually works.

GET IN TOUCH
contact@lumiops.ai

CATALOG
32 built-in skills

LINKEDIN
linkedin.com/lumiops