

— THE COMPLETE GUIDE

The Automation Library Playbook.

How to stop rewriting the same runbook for every site, pick the right machinery for each job, and build a library of automations your team trusts because the numbers say they work.

12 chapters 3 frameworks 3 automation types

— THREE FRAMEWORKS YOU CAN REUSE

01

The Machinery Match

Five questions that pick workflow, agentic workflow or agent.

02

The Fork Test

When to fork, when to tune, when to build from blocks.

03

The Library Health Check

Ten checks for a library people actually trust.

— WHO THIS IS FOR

- Platform teams maintaining a script per site
- Ops leads who inherited three versions of everything
- Engineers tired of starting from a blank canvas

— WHAT THE LIBRARY IS BUILT ON

3

AUTOMATION TYPES

10

BLOCK FAMILIES

1-click

RUN ON ANY SITE

18

PLATFORMS TARGETED

Fixing a procedure once should fix it everywhere it runs.

Build once, run anywhere — one library of workflows, agentic workflows and agents for every site you run.

— CONTENTS

What this playbook covers

This is for teams whose automation grew one script at a time and now lives in six places. It assumes you already automate plenty. The question it answers is how to turn that pile into one library: the right machinery for each job, one definition per procedure, and a score that tells you what to trust.

01 Why teams keep rewriting the same runbook The copy problem	p03 02 One definition, many sites Why the site should decide	p04
03 Framework 1 — the Machinery Match Workflow, agentic workflow or agent	p05 04 Deterministic workflows When the path must not vary	p06
05 Agentic workflows Reasoning inside the rails	p07 06 Agents Standing specialists	p08
07 Framework 2 — the Fork Test Fork, tune or build	p09 08 The block palette Triggers to Terraform	p10
09 Success rate as the trust signal Promote what works	p11 10 Personal, org, curated Where automations live	p12
11 Framework 3 — the Library Health Check Ten checks for a library you can trust	p13 12 Getting started Your first forked automation	p14

— THREE FRAMEWORKS YOU CAN REUSE

The Machinery Match

Five questions that pick workflow, agentic workflow or agent.

The Fork Test

When to fork, when to tune, when to build from blocks.

The Library Health Check

Ten checks for a library people actually trust.

— BRING THESE TO THE FIRST READ-THROUGH

- A list of every script and automation you run today
- Which sites each one targets
- Who wrote it, and whether they still work here
- Which ones anyone actually trusts

— WHAT THE LIBRARY IS BUILT ON

3

AUTOMATION TYPES

10

BLOCK FAMILIES

1-click

RUN ON ANY SITE

18

PLATFORMS TARGETED

You do not need to automate more. You need to automate once.

— CHAPTER ONE

Why teams keep rewriting the same runbook

Nobody sets out to maintain four patch scripts. It happens one reasonable decision at a time — a new region, a new cloud, a quick fix that never got merged back.

The copy problem is sneaky because each copy starts out correct. The datacenter version works. The cloud version is adapted from it and also works. Then someone fixes a bug in one and not the others, a new step gets added to two of the three, and within a year you have several procedures with the same name that do slightly different things. Nobody decided that. It just accumulated.

— LISTEN FOR THESE PHRASES

SYMPTOM	WHAT IT REALLY MEANS
"Use the new script, not the old one"	Two versions, no source of truth
"That one only works in eu-west"	Site detail baked into the procedure
"Ask Sam, Sam wrote it"	Ownership lives in a person, not a library
"It usually works"	Nobody is measuring success

Every copy of a runbook is a future disagreement about which one is right.

— THE SHAPE OF THE PROBLEM (ILLUSTRATIVE)

4+

COPIES OF ONE PROCEDURE, TYPICAL

0

OF THEM SCORED ON SUCCESS

1

DEFINITION YOU ACTUALLY NEED

— PILE VS LIBRARY

A PILE OF SCRIPTS

- Found by asking around
- Fixed in one place at a time
- Trusted by reputation
- Owned by whoever wrote it

A LIBRARY

- Found by searching
- Fixed once, everywhere
- Trusted by success rate
- Owned by a named reviewer

— CHAPTER TWO

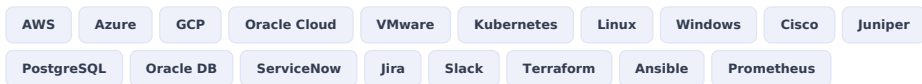
One definition, many sites

The fix for the copy problem is simple to state: the procedure lives in one place, and the site you point it at decides where it lands.

— SPLITTING PROCEDURE FROM PLACE

BELONGS IN THE AUTOMATION	BELONGS IN THE SITE
The steps and their order	Hostnames, clusters, regions
The checks that prove it worked	Credentials and connection details
Branches and approvals	Maintenance windows local to the site
What gets reported afterwards	Which platform API to call

— ONE DEFINITION CAN TARGET ANY OF THESE



Once the split is clean, adding a new site stops being a porting exercise. It inherits every proven automation on day one, and any fix you make to the procedure lands everywhere at once.

If a workflow has a region name in it, it isn't finished yet.

— WHAT THE SITE CARRIES

```
site: plant-04-vmware
maintenance_window: Sun 01:00–04:00 local
targets: tag role:line-controller
credentials: vault://plant-04/patch
# the workflow never mentions plant-04 – the site does
```

• New site = new config, not a new workflow.

• A fix to the procedure reaches every site at once.

Portability is not a feature you add later. It is what happens when the procedure stops knowing where it runs.

— CHAPTER THREE · FRAMEWORK ONE

The Machinery Match

Five questions that decide whether a job wants a fixed workflow, an agentic workflow, or a standing agent. Answer them before you build anything.

QUESTION	IF YES, LEAN TOWARDS
Is the procedure fully known and must it never vary?	Workflow
Does it need diagnosis or judgement at one or two points?	Agentic workflow
Should it run continuously and watch for something?	Agent
Would a wrong branch be expensive?	Agentic workflow with an approval on that branch
Will people direct it in plain language?	Agent

— THE THREE OPTIONS

WORKFLOW	AGENTIC WORKFLOW	AGENT
Deterministic Fixed steps, the same path every time. For procedures that are already known and must not vary. on schedule → query CMDB → run script → post to Slack	Reasoning inside the rails A structured flow with AI blocks that diagnose, branch and summarise — bounded by the steps you defined. on alert → AI diagnose → branch on confidence → remediate or escalate	Standing, always-on A persistent specialist you direct in plain language. It picks its own path, inside its guardrails. "find IAM policies that drifted this week" → investigate → report

Use the least clever machinery that does the job. Save reasoning for the parts that genuinely need it.

— A WORKED EXAMPLE

JOB	KNOWN?	JUDGEMENT?	CONTINUOUS?	MATCH
Certificate renewal	Yes	At one branch	Daily	Agentic workflow
Patch window	Yes	No	Scheduled	Workflow
IAM drift watch	Mostly	Yes	Always	Agent
Crash-loop triage	Partly	Yes	On alert	Agentic workflow

— CHAPTER FOUR

Deterministic workflows

A workflow is the most boring option in the library, and that's its strength. Same steps, same order, same checks, every time. When the procedure is known, boring is exactly right.

— WHAT A GOOD WORKFLOW READS LIKE

```
patch-window-runner workflow
  trigger: schedule · Sun 01:00 (site-local window)
  step: drain host from service
  step: apply patches
  step: reboot and health-check
  step: return host to service
  output: summary to #ops, ticket updated
```

— JOBS THAT SUIT A WORKFLOW

- Certificate renewal
- Patch windows
- Backup restore drills
- Config drift rollback to a golden baseline
- Terraform plan, approve, apply
- Nightly report exports

SIGNS IT SHOULD STAY A WORKFLOW

- The runbook has no "it depends"
- Every run should look identical
- Auditors care about exact steps
- Failure means stop, not improvise

SIGNS IT HAS OUTGROWN ONE

- Engineers keep adding special cases
- The same step needs judgement each time
- Half the runs escalate anyway
- People edit it before every run

— EVERY STEP SHOULD PROVE SOMETHING

STEP	WHAT IT MUST PROVE BEFORE MOVING ON
Drain host	No live sessions or jobs on the host
Apply patches	Package manager reports success
Reboot and health-check	Service checks green within the time limit
Return to service	Traffic flowing and error rate normal

— CHAPTER FIVE

Agentic workflows

An agentic workflow keeps the structure you trust and adds AI blocks exactly where judgement is needed — to diagnose, to decide which branch to take, to summarise what happened.

— REASONING, BOUNDED BY YOUR STEPS

cert-renewal-prod

agentic workflow

```
trigger: schedule · daily 02:00
query: certificates expiring within 60 days
ai: rank by blast radius, draft renewal plan
branch: confidence ≥ 0.9 → renew · else → escalate
action: renew → deploy → validate TLS handshake
output: post what changed to #platform, attach to ticket
```

— THE AI BLOCKS, AND THEIR RAILS

AI BLOCK	WHAT IT DOES	WHAT BOUNDS IT
Diagnose	Reads the evidence and proposes a cause	The query sources you connected
Branch on confidence	Chooses the path based on how sure it is	A threshold you set; below it, escalate
Summarise	Writes up what changed, for humans	The output block and its destination

The structure is the guardrail. The AI decides inside it — never around it.

— USE REASONING WHERE IT EARNS ITS KEEP

WHERE AI BLOCKS HELP

- Reading messy evidence
- Choosing between known branches
- Writing the summary humans read

WHERE THEY GET IN THE WAY

- Steps that must be identical every run
- Changes auditors need to see literally
- Anything a threshold cannot bound

Reasoning inside rails is still reasoning — so give every AI branch a threshold and a way to say "I am not sure".

— CHAPTER SIX

Agents

An agent is a standing specialist. You direct it in plain language, and it chooses its own path — within its guardrails. It's the right tool for watching, not for one-off change.

— DIRECTING AN AGENT

```
direct: "find IAM policies that drifted from the repo this week"
agent: iam-drift-scanner · standing
  → compares live policies with source of truth
  → finds 3 drifted roles in two accounts
  → reports diffs and likely cause
✓ report posted · no changes made
```

— WHAT AGENTS ARE GOOD AT

IAM drift

Finds policies that no longer match source control.

Spend anomalies

Surfaces services drifting from their 14-day baseline.

Disk investigation

Watches volumes against your own cleanup policy.

— USING AGENTS WELL

- Give it a clear standing job, not a vague mandate.
- Watch its success rate like any other automation.
- Keep one-off changes in workflows, not agents.
- Fork a prebuilt agent and tune it to your layout.

— ONE JOB, TWO MACHINES

SAME JOB	AS A WORKFLOW	AS AN AGENT
Spend review	Weekly report, fixed query	Always watching the 14-day baseline
IAM hygiene	Monthly diff against repo	Directed: "what drifted this week?"
Disk cleanup	Scheduled delete by policy	Investigates first, reports before acting

— WHAT EVERY AGENT NEEDS BEFORE IT GOES LIVE

- A clear standing job, stated in one sentence.
- A report destination people actually read.
- Guardrails that match the systems it watches.
- A success rate reviewed like any other automation.

— CHAPTER SEVEN · FRAMEWORK TWO

The Fork Test

Before you build anything from blocks, check whether you should. Most of the time, forking something that already works is faster and safer.

1 Run it as-is. A prebuilt automation already matches your procedure and platforms.
Choose this when: it passes a dry run on one of your sites

2 Fork and tune. It covers most of the job; your procedure differs in a step or two.
Choose this when: you can name exactly which steps differ

3 Build from blocks. Nothing in the library is close, or the procedure is truly yours.
Choose this when: you checked the library first — properly

— WORKED EXAMPLES

SITUATION	MOVE
Same procedure, different site	Run as-is — point it at the new site
Same job, your cleanup policy differs	Fork and tune the relevant step
Org-specific triage using your own metrics	Fork, then wire it to your systems
A procedure nobody else has	Build from blocks

A fork keeps the original intact. You get your version, and the curated one keeps improving for everyone.

— WHAT A FORK ACTUALLY CHANGES

```
fork my-disk-investigator ← forked from UnityOne disk investigator
+ step: respect /data/archive retention (90 days)
+ target: our volume layout (lvm, /srv split)
~ output: post to #storage-ops instead of email
original: unchanged · still receives curated updates
```

• Name the steps that differ before you fork.

• Keep the fork linked to its source.

— CHAPTER EIGHT

The block palette

When you do build, you build from block families that map to real operational work — from what starts the automation to where the result ends up.

01

Triggers

Schedule, on alert, webhook, or kicked off by hand.

02

Advanced AI blocks

Diagnose, decide, branch and summarise.

03

Task actions

The operational steps themselves.

04

Flow controls

Branches, loops, waits and approvals.

05

Query sources

CMDB, monitoring, logs, cloud APIs.

06

Run commands

Shell and remote execution on targets.

07

Terraform & scripts

Plan, apply, and your own scripts.

08

HTTP / API actions

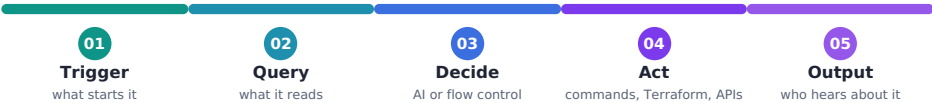
Call anything with an endpoint.

09

Outputs

Slack, tickets, reports, email.

— MOST AUTOMATIONS FOLLOW THE SAME SHAPE



If you can't say what triggers it and where the output goes, you're not ready to build the middle.

— FAMILIES IN PRACTICE

FAMILY	AN EXAMPLE BLOCK
Query sources	Certificates expiring within 60 days
Flow controls	Wait for approval from on-call lead
Terraform & scripts	terraform plan, post diff
Outputs	Summary to Slack, ticket updated

— CHAPTER NINE

Success rate as the trust signal

Reputation is a terrible way to decide which automation to trust. A live success rate on every row replaces 'Sam says it's fine' with a number that updates every run.

— A SAMPLE LIBRARY, RANKED BY SUCCESS



— READING THE NUMBER

SCORE	WHAT TO DO
95% and up	Candidate for org-wide sharing and scheduled runs
85-94%	Useful; look at the failures for a common cause
Below 85%	Fix before relying on it — or narrow what it is pointed at

Scores shown are from a sample library and are illustrative. The thresholds are a starting point — set your own.

— WHAT THE NUMBER REPLACES

TRUST BY REPUTATION

- "Sam says it is fine"
- Failures remembered selectively
- Old scripts never retired

TRUST BY SCORE

- A number on every row
- Every run counted, good and bad
- Low scorers fixed or removed

— CHAPTER TEN

Personal, org, curated

Where an automation lives says a lot about how much to trust it. The library makes that explicit, so nobody runs a personal experiment on a production schedule by accident.

Personal**Private to you**

Drafts and experiments. Build freely; promote to the org once it proves out.

Org**Shared with your workspace**

Your team's runbooks, wired to your systems and reviewed by your people.

UnityOne**Prebuilt and curated**

Production-tested automations for common ops work. Fork and tune without touching the original.

- 1 Personal.** Draft freely. Nothing here runs on shared schedules.

Promote when: it works on your own test runs

- 2 Org.** Shared with the workspace, wired to your systems.

Promote when: its success rate holds on real sites

- 3 Scheduled & relied on.** Running on a cadence the team depends on.

Promote when: the team agrees the score is good enough

— HOUSE RULES WORTH ADOPTING

- Promote on evidence, not enthusiasm.
- Org automations should have a named reviewer.
- Forks remember where they came from.
- Retire duplicates as soon as one version wins.

— VISIBILITY, AT A GLANCE

SOURCE	WHO CAN SEE IT	WHO CAN RUN IT
Personal	Only you	Only you
Org	The whole workspace	Anyone your roles allow
Forked from UnityOne	Wherever the fork lives	By where it lives

A personal draft on a production schedule is how good intentions become incidents.

- Share to the org only once the score holds on real sites.
- Retire the personal copy after sharing.

— CHAPTER ELEVEN · FRAMEWORK THREE

The Library Health Check

Ten questions for a library people trust. Run it quarterly — libraries drift the same way scripts do if nobody looks.

01

Is there exactly one definition per procedure?

02

Do workflows avoid site-specific details?

03

Is each automation the least clever type that does the job?

04

Does every AI branch have a confidence threshold?

05

Does every risky branch have an approval?

06

Is a success rate visible on every row?

07

Are low scorers being fixed, not ignored?

08

Do forks point back to their source?

09

Does every org automation have a named reviewer?

10

Have duplicates been retired?

— SCORING IT

1**Eight or more**

A library people can rely on.

2**Five to seven**

Healthy, with a clean-up list.

3**Four or fewer**

Still a pile of scripts. Start with duplicates.

The best time to delete a duplicate is the day the better version wins.

— WHAT THE CHECK USUALLY FINDS

COMMON FAILURE	USUAL FIX
Two automations, one procedure	Keep the higher scorer; retire the other
Region names inside a workflow	Move them into site config
AI branch with no threshold	Add one; escalate below it
Nobody reviews org automations	Name a reviewer per category

— CHAPTER TWELVE

Getting started

You don't need to migrate everything. Pick the procedure with the most copies, fork the closest prebuilt automation, and run it on one site this week.

STEP	WHAT TO DO	WHAT YOU GET
1 · Count the copies	List procedures that exist in more than one place	Your best first candidate
2 · Search the library	Look for a prebuilt match first	Most of the work already done
3 · Run the Machinery Match	Workflow, agentic workflow or agent?	The right tool for the job
4 · Fork and point	Tune the differences; choose one site	A working automation, fast
5 · Watch the score	Let a week of runs build a success rate	Evidence, not opinion
6 · Promote and retire	Share it with the org; delete the old copies	One definition, everywhere

— YOUR FIRST WEEK, ROUGHLY

1 Monday Count copies of your procedures.	2 Tuesday Search the library; fork the closest match.	3 Wednesday Point it at one site and run it.	4 Thu-Fri Watch the score; plan the retirements.
---	---	--	--

— FIVE THINGS WORTH TAKING AWAY

- The copy problem is the real enemy, not a lack of automation.
- Use the least clever machinery that works.
- Trust the score, not the reputation.
- One definition; the site decides where it lands.
- Fork before you build.

Ready to fork your first automation?

Search the library, point it at one site, and let a week of runs tell you how good it is.

Start with the procedure that has the most copies. It will teach you everything else in this playbook.

— IN CLOSING

Build it once. Run it anywhere.

Your team has already written most of the automation it needs — several times over. The work now is choosing one version of each, putting it in a library, and letting the numbers decide what everyone relies on.

3

AUTOMATION TYPES

10

BLOCK FAMILIES

1-click

RUN ON ANY SITE

18

PLATFORMS TARGETED

— THE THREE FRAMEWORKS, IN ONE PLACE

The Machinery Match

Five questions that pick workflow, agentic workflow or agent.

The Fork Test

When to fork, when to tune, when to build from blocks.

The Library Health Check

Ten checks for a library people actually trust.

— THE WHOLE PLAYBOOK, IN THREE WORDS

1**Count**

Find the procedure with the most copies.

2**Fork**

Start from the closest prebuilt automation.

3**Prove**

Let the success rate decide.

The goal isn't a bigger library. It's one you'd bet the maintenance window on.

- One definition per procedure.
- The least clever machinery that works.
- Fork before you build.
- Trust the score.

Stop rewriting the same runbook.

Fork a prebuilt automation, point it at your site, and run it today.

GET IN TOUCH

contact@lumiops.ai

RUN

1-click, any site

LINKEDIN

linkedin.com/lumiops