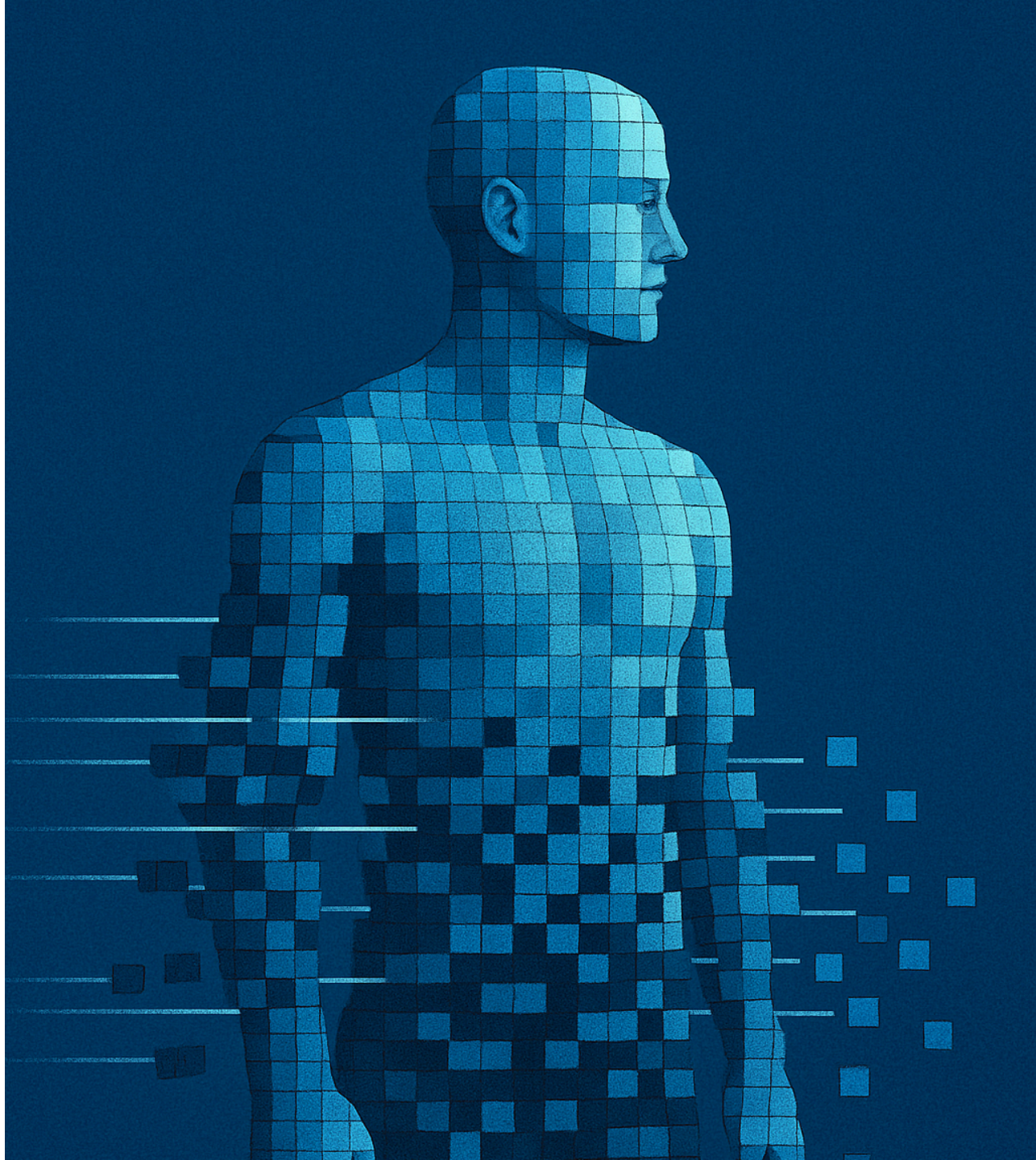
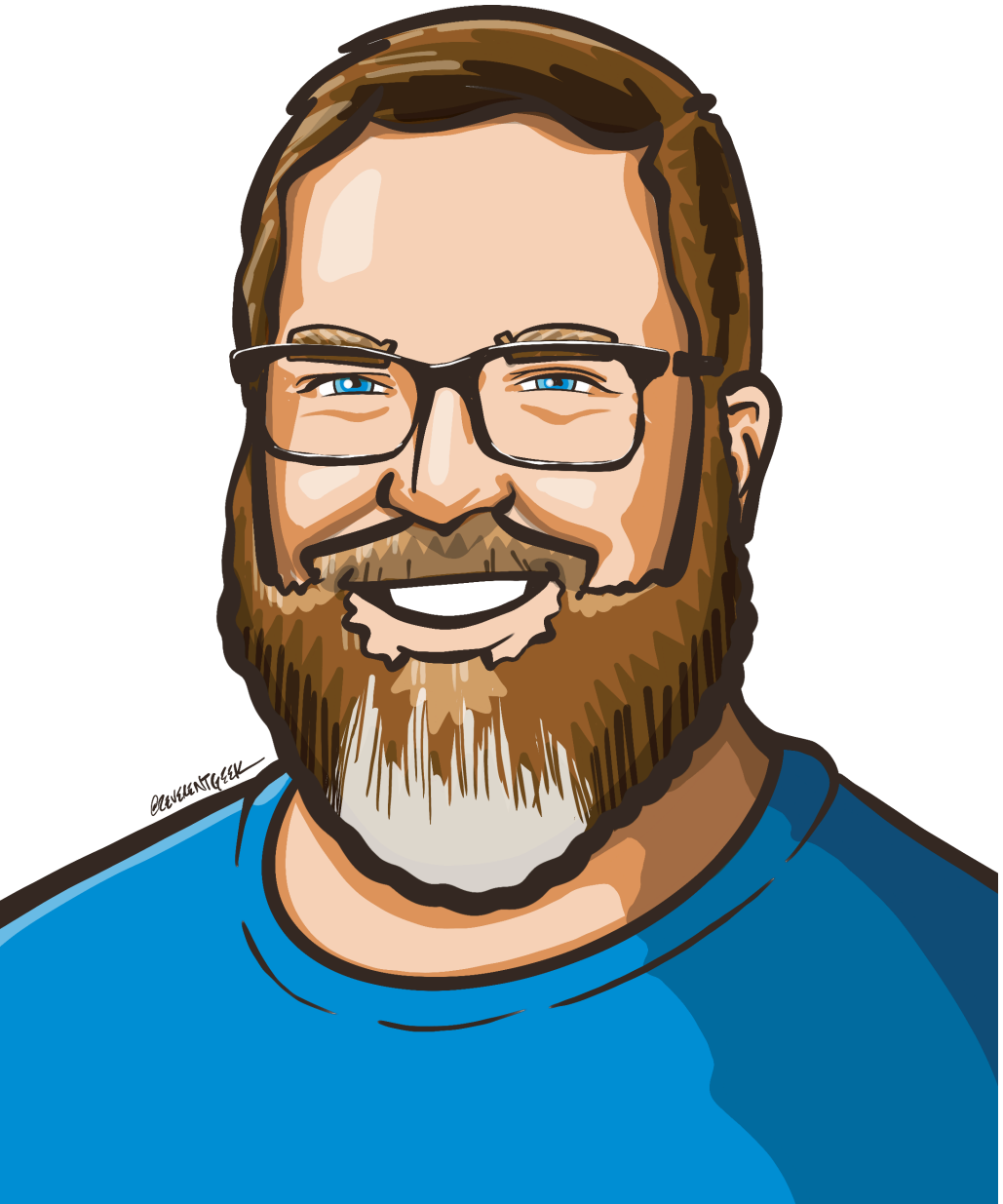
Refactoring Your Identity

Keep your expertise. Expand where it works.

Chris Ayers





Chris Ayers

Principal Software Engineer

Azure EngOps AzRel

Microsoft

Bluesky: @chris-ayers.com

LinkedIn: chris-l-ayers

Blog: https://chris-ayers.com/

GitHub: Codebytes

Mastodon: @Chrisayers@hachyderm.io

~~Twitter: @Chris_L_Ayers~~

🧵 Coding Kata Meetup

> *I'm not a _____ developer. I'm a XXXXXXXXXXXX Developer.*

I'm stack-agnostic. As long as it's my stack.

Who are you?

Have you ever passed on work because you weren't **"that kind" of developer?**

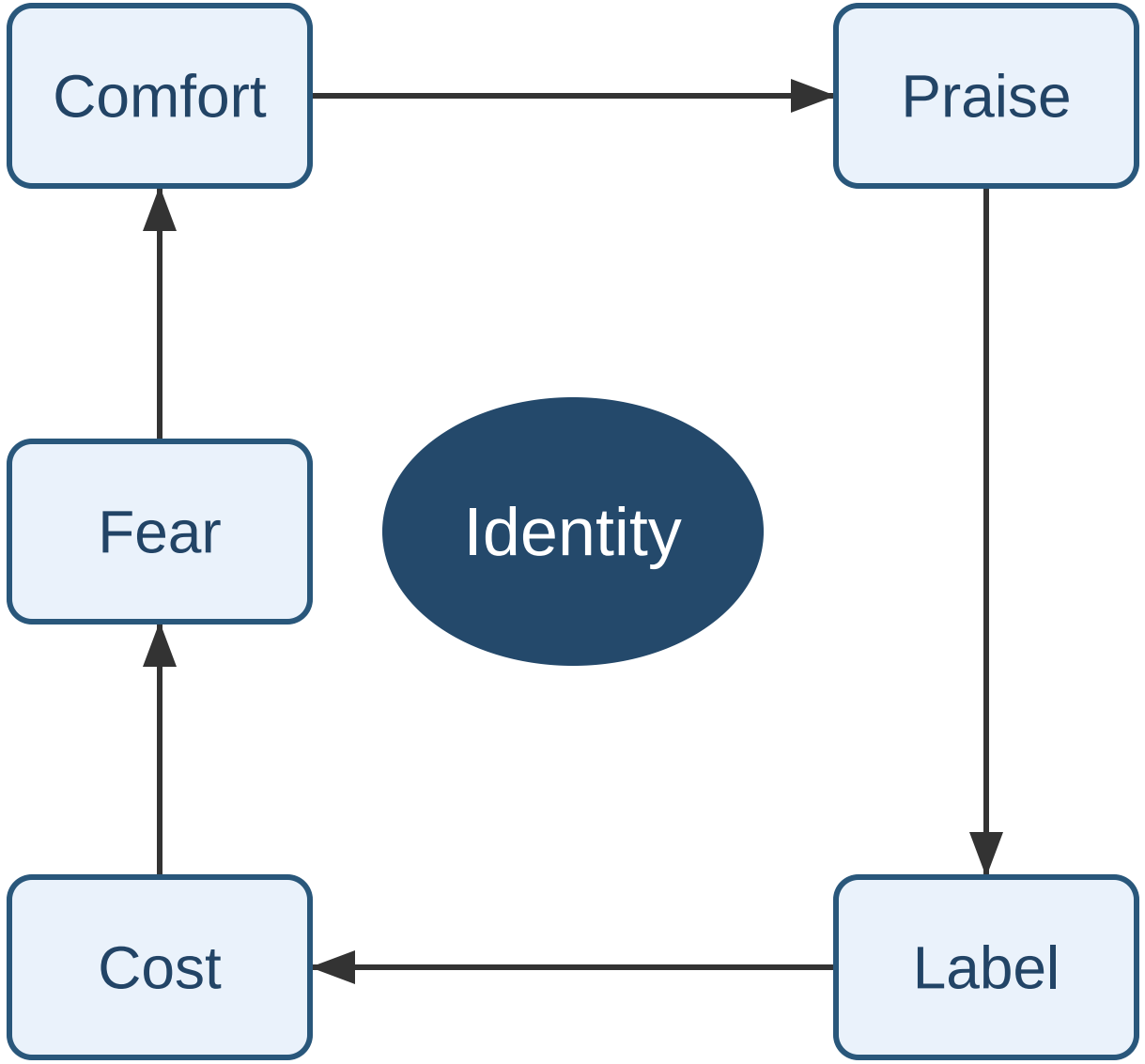
🌱 One Strength. One Experiment.

- Find a skill you can use somewhere unfamiliar
- Choose one small experiment for the next 30 days
- Leave with a five-line record you can reuse


How did we get here?

How Technical Identity Forms

- Early wins build confidence
- You get faster and receive praise
- Work comes back to **"The Expert"**
- Repetition builds comfort and narrows your range



When a Label Makes the Choice

Tool first

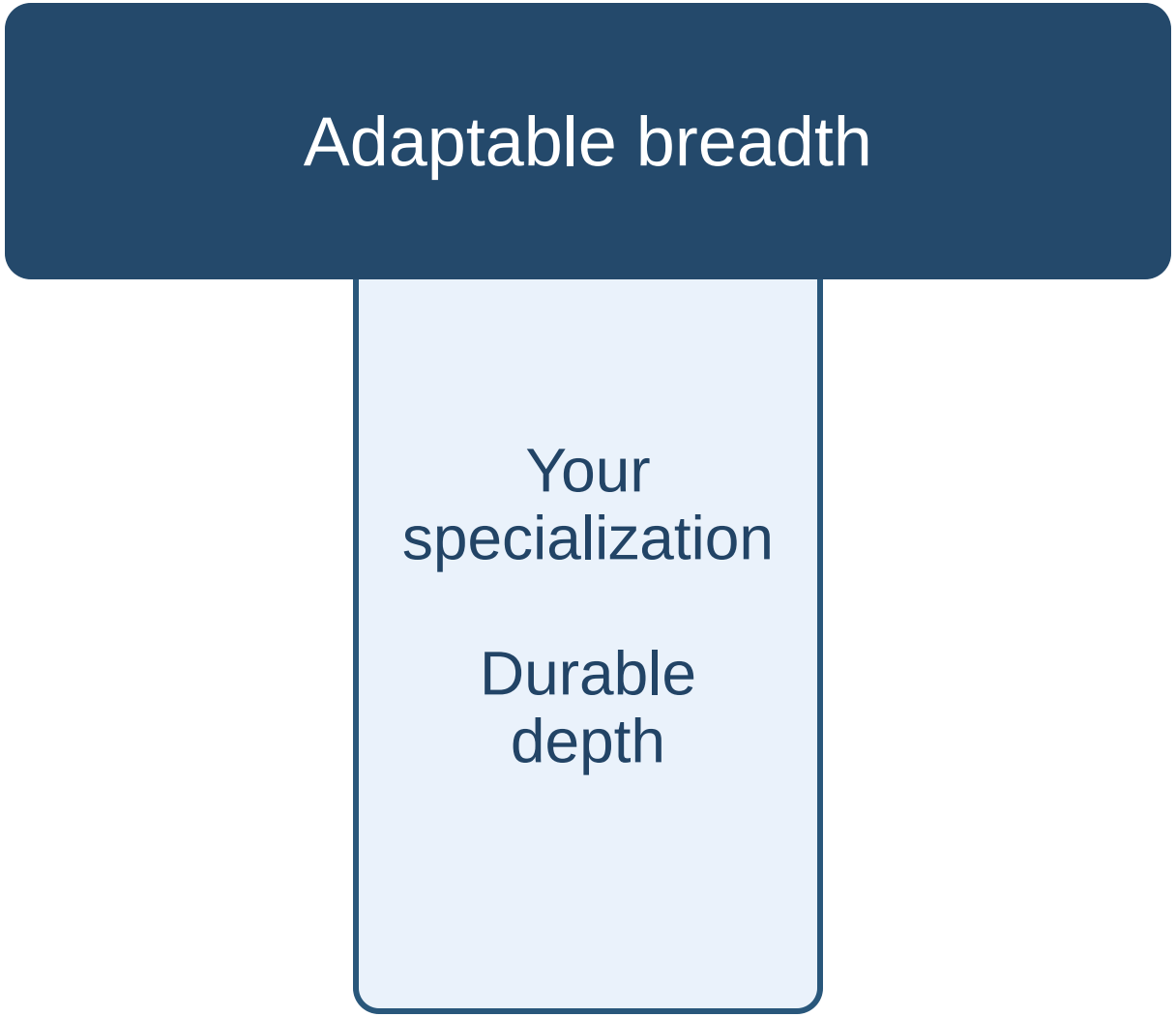
Pick the familiar solution before considering the requirements.

Status first

Avoid work where someone might see you need help.

Comfort first

Rule out an opportunity because it is outside your stack.

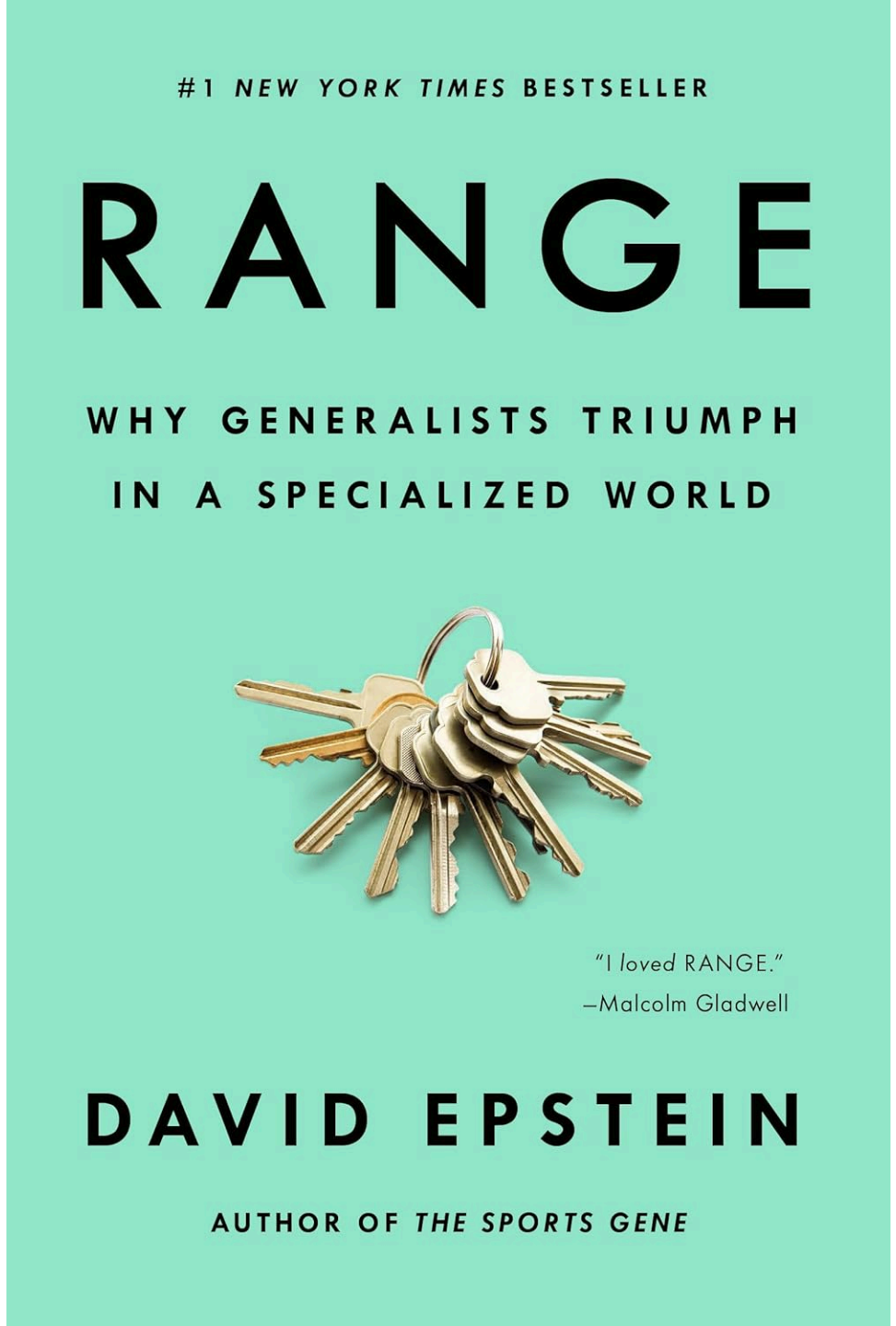


↓ Depth Is Not the Problem

- Keep your specialization
- Make room for work outside it
- Think **T-shaped**: deep expertise with room to branch out

Range & Generalists

- David Epstein's *Range- makes the case for breadth in complex, changing work
- Experience elsewhere can help you recognize a useful pattern
- Build on your depth while trying something outside it



Pick One Example to Carry Forward

Write down	Your example
One avoided area	A stack, domain, or kind of work
One concern	Time, status, uncertainty, or needing help
One existing strength	Something you could bring to that work

Keep this note. We will turn it into an experiment.

Share & Normalize (Optional)

- Share one area you've avoided
- Listen for a pattern you recognize
- Keep the personal parts private if you prefer

Refactor Your Identity

*Before- It Hardens

Chris Ayers | https://chris-ayers.com

What Repetition Can Cost

Fewer possibilities

- The familiar choice becomes the default
- You get less practice comparing approaches

Less room to learn

- Maintenance can fill the available time
- Defending a niche can become exhausting

🧵 My Re-Org: FTA to AzRel

Customer architecture

- Help customers adopt Azure
- Design and build cloud-native solutions

Reliability engineering

- Learn risks, outages, and new tools
- Build AI-assisted analysis with other teams

New context. Existing habits: ask, map, explain.

Refactor Your Identity *Intentionally*

Chris Ayers | https://chris-ayers.com

⚠ Trigger Moments to Watch For

⚡ You feel slower

- Is the work unfamiliar, or is the approach wrong?

🧭 The problem is unclear

- Are you picking a tool before asking enough questions?

🛡 Risk gets deferred

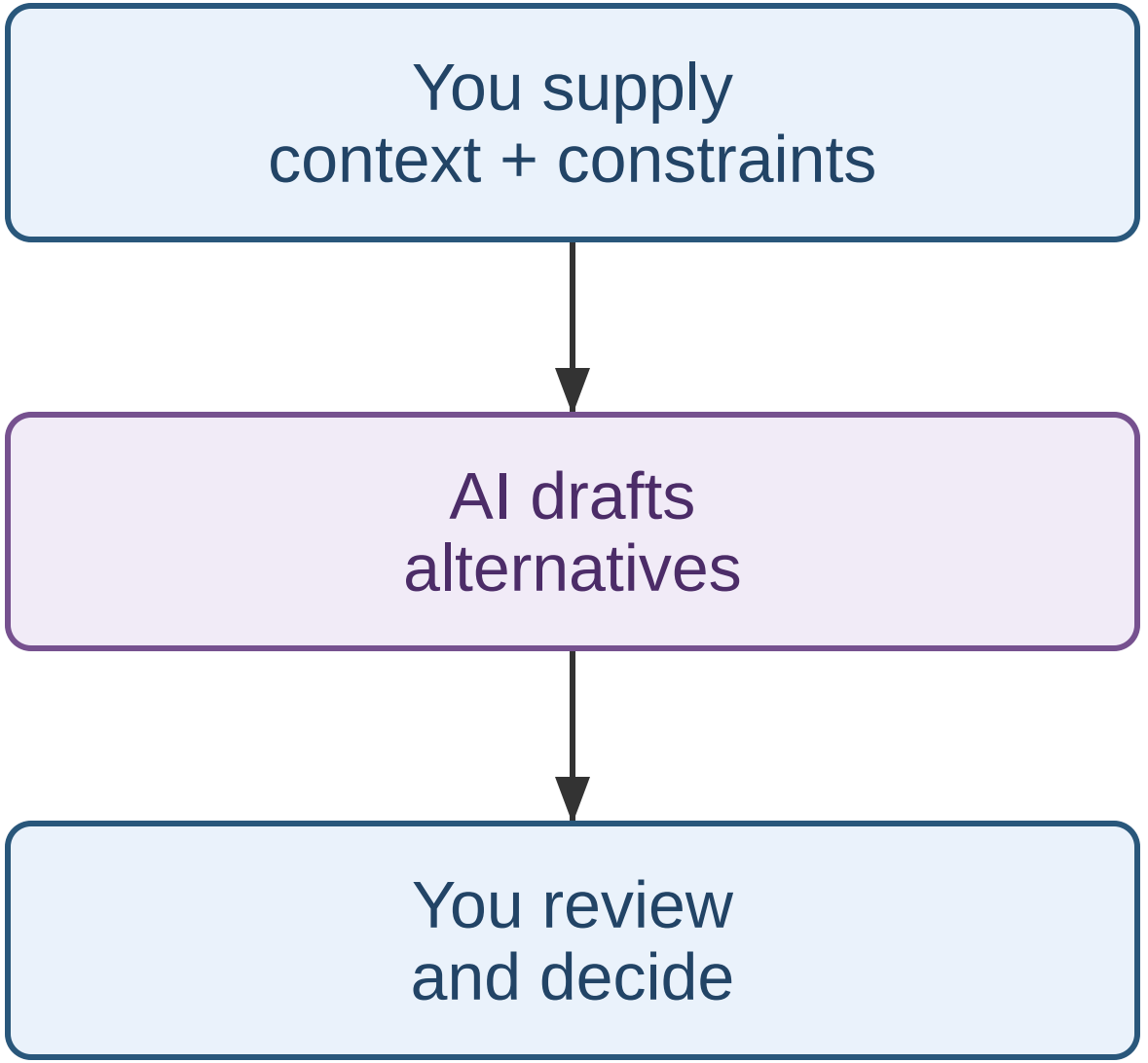
- Are you forcing old patterns into a different context?

Small Interventions at the Moment of Choice

- Spend 2 minutes considering another option
- Ask what has changed since the last decision
- Choose the smallest useful experiment
- Capture one decision in 5 lines

🪄 AI Lowers the Cost of Trying

- Map unfamiliar code before editing it
- Compare patterns with what you already know
- Try a small change you can explain and test



🦆 My Rubber Duck Has Opinions Now

Give it context

- The problem and constraints
- What an acceptable result looks like

Challenge the answer

- Alternatives and failure cases
- Evidence that the result works

Direct AI. Don't just use it.

What You Still Own

- Deciding which problem to solve
- Choosing trade-offs under real constraints
- Checking whether the output is correct and safe
- Getting people to agree on a direction

If you cannot explain the change, keep investigating.

Ask for a Map Before a Patch

Example: a form in an unfamiliar codebase.

> *Trace the Save action from the form to the API. Cite the files you used.*
>
> *Separate observations from assumptions. Suggest the smallest check for each uncertainty.*
>
> *Don't change code yet.*

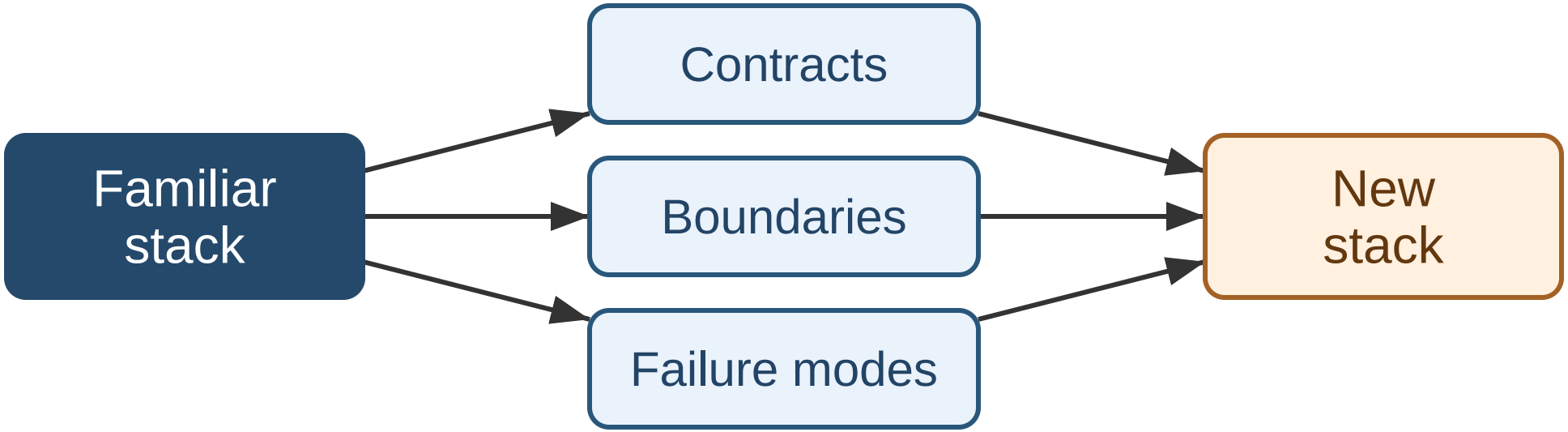
Read the code. Run the checks. Explain the result.

Refactor Your Identity *For Growth*

Chris Ayers | https://chris-ayers.com

🧵 Cross-Stack Win

Backend habits. Front-end problem.



> *The stack was unfamiliar. The thinking was portable.*

A Contact Saved Twice

Example: you're new to the team's UI framework.

1. A user enters a contact and clicks **Save**.
2. No confirmation appears. They click **Save** again.
3. The contact appears **twice** in the list.

A retry should not create a second record.

Three Questions. Eight Portable Skills.

Judgment

What fits the outcome?

Systems design
Business value
Trade-offs

Investigation

What does the evidence say?

Problem solving
Debugging discipline
Quality & governance

Shared impact

Who can use the reasoning?

Communication
Mentorship & leadership

Judgment: What Must Stay True?

Customer outcome: one record for one intended submission.

Boundary	Contract
Intent and retry	Same intent, same operation key
API and storage	Enforce one result for that operation
Result and UI	A timeout is not proof of failure

⚖️ Trade-Offs: Which Fix Fits?

Disable the button

- Improves feedback during a request
- Does not make network retries safe

Enforce idempotency

- Protects retries of the same operation
- Adds server-side state and policy

Which protects a retry after a lost response?

Investigation: Follow the Evidence

- **Hypothesis:** one intent reached the write path twice
- **Evidence:** correlate operation keys, requests, and writes
- **Experiment:** delay the response after a write, then retry

What would distinguish a double-click from a network retry?

Make the Failure Safe and Visible

Before rollout

Exercise timeouts, retries, and concurrent requests.

At the boundary

Authorize the caller. Scope keys to the caller and operation.

After rollout

Observe duplicate attempts, successes, and failures.

💬 Shared Impact: Leave the Reasoning

Explain

- Check the shared meaning of "one submission"
- Explain the options, constraints, and decision

Enable

- Share the reproduction and decision record
- Let someone else replay and challenge them

Five Lines Others Can Reuse

Line	Record
Context	A retry can create a duplicate record.
Options	UI guard alone; server-enforced idempotency.
Criteria	Safe retries and an operable change.
Decision	Enforce idempotency; a UI guard alone misses retries.
Evidence to collect	One result under retries and concurrent requests.

Now Apply It to Your Example

What stays?

The existing strength you wrote down.

What changes?

The stack, domain, or kind of work.

What will you try?

One small way to use that strength there.

You do not need fluency before you can ask useful questions.

Chris Ayers | https://chris-ayers.com

Start Where You Are

- Keep your current job; change what you practice
- Borrow a problem from your team, community, or open source
- Pick one small experiment for the next 30 days

Your 30-Day Focus

Pick one starting action. Not all four.

Choose	Try
Notice a strength	Name a strength and somewhere new to use it.
Record a decision	Capture five lines beside the work.
Check an assumption	Clarify one in a meeting, 1:1, or study group.
Run an experiment	Try something unfamiliar; record what you learned.

Your 60-Day Follow-On

Extend the habit you started. Choose one follow-on.

Choose	Try
Review your balance	Pick an underused skill from the last month's work.
Combine strengths	Apply two skills to one real problem.
Teach something	Share for 10 minutes; keep two follow-up questions.
Review weekly	Notice what helped and adjust your next experiment.

Who Are You Beyond the Stack?

> *I help **[people]** achieve **[outcome]**.*
> *I bring **[portable strength]**.*
> *Right now, I work in **[stack]**.*

Choose your experiment before you leave.

Thank You

Refactor Your Identity

Continuously. Not Reactively.

One strength. One experiment.

Resources

Range - David Epstein

25 Transferable Skills Employers Look For

Chris Ayers

Principal Software Engineer

Azure EngOps AzRel

Microsoft

Bluesky: @chris-ayers.com

LinkedIn: chris-l-ayers

Blog: https://chris-ayers.com/

GitHub: Codebytes

Mastodon: @Chrisayers@hachyderm.io

~~Twitter: @Chris_L_Ayers~~