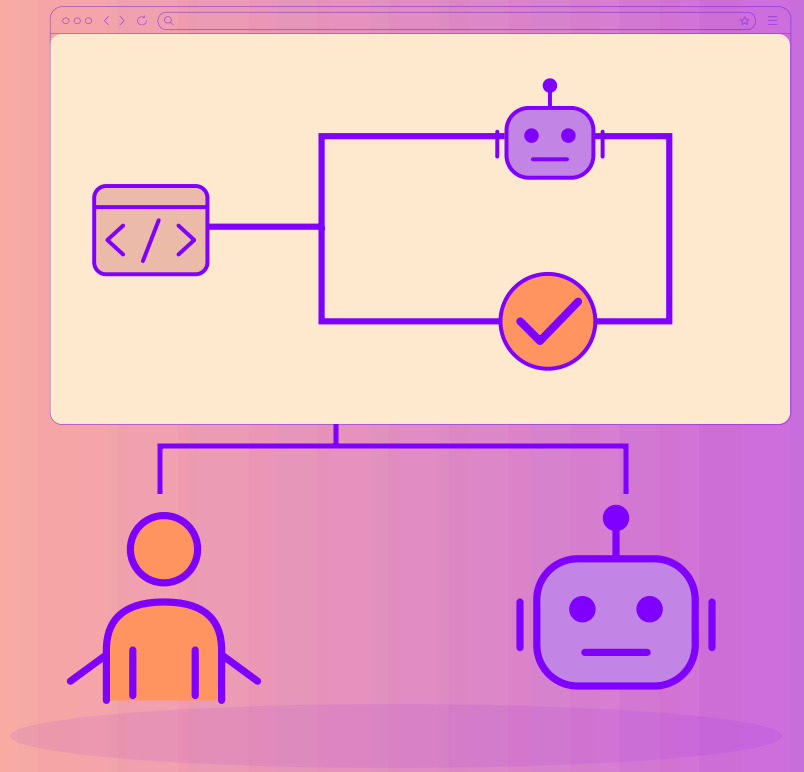
From Saw to Chainsaw

Without Losing a Hand

Coaching teams to own agentic workflows



João Anes

(pronounced joe-WOW)

joao.anes@mindera.com

Confidential

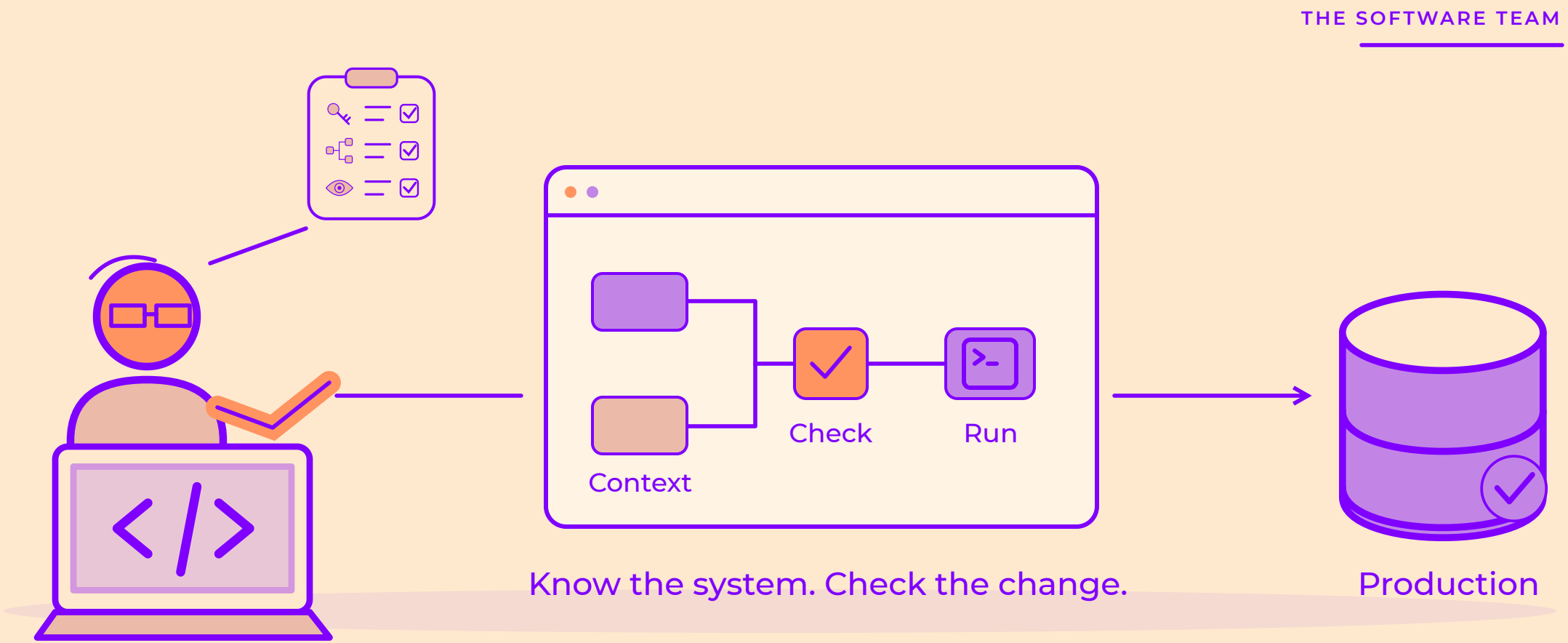
© 2026 Mindera

Let me tell you a story about **woodworking.**

Confidential

© 2026 Mindera

Engineers have been doing this for years.

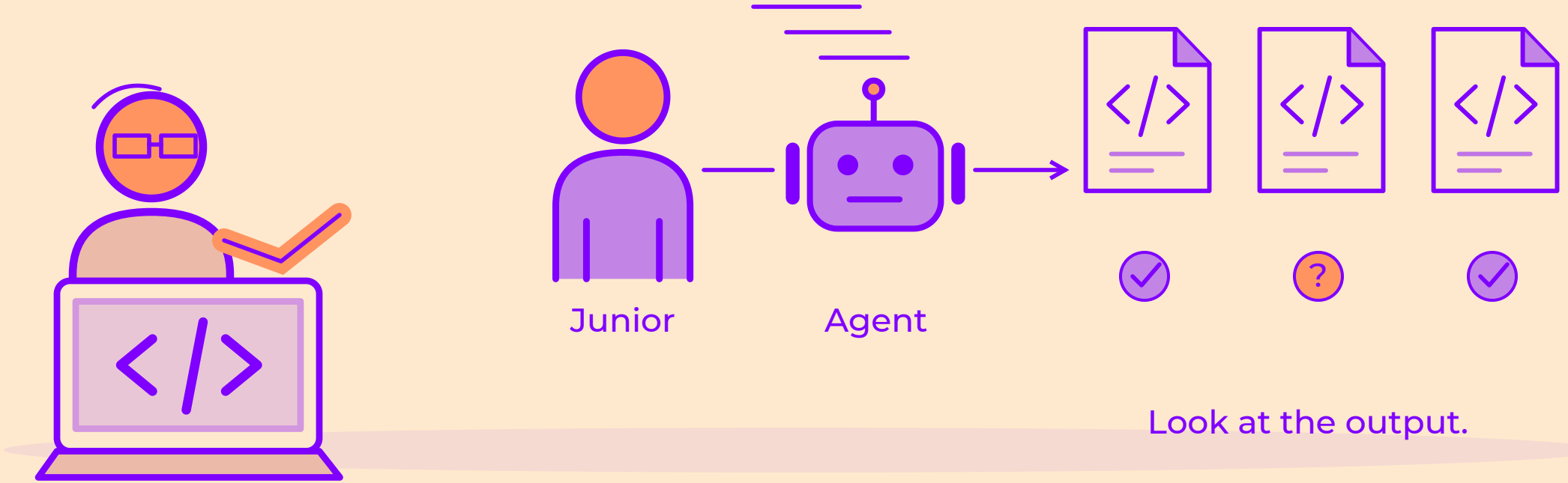


Confidential

© 2026 Mindera

Here comes new technology.

THE SOFTWARE TEAM



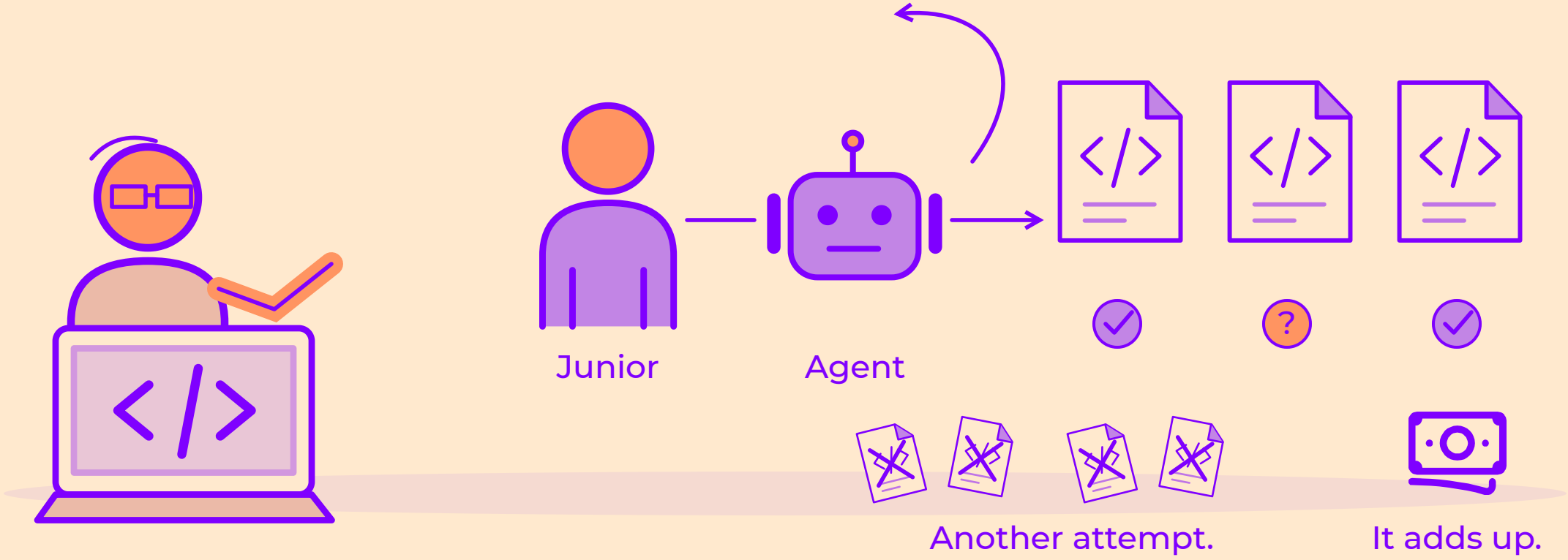
Look at the output.

Confidential

© 2026 Mindera

We can see a lot of waste.

THE SOFTWARE TEAM



Confidential

© 2026 Mindera

Someone still has to check it.

THE SOFTWARE TEAM

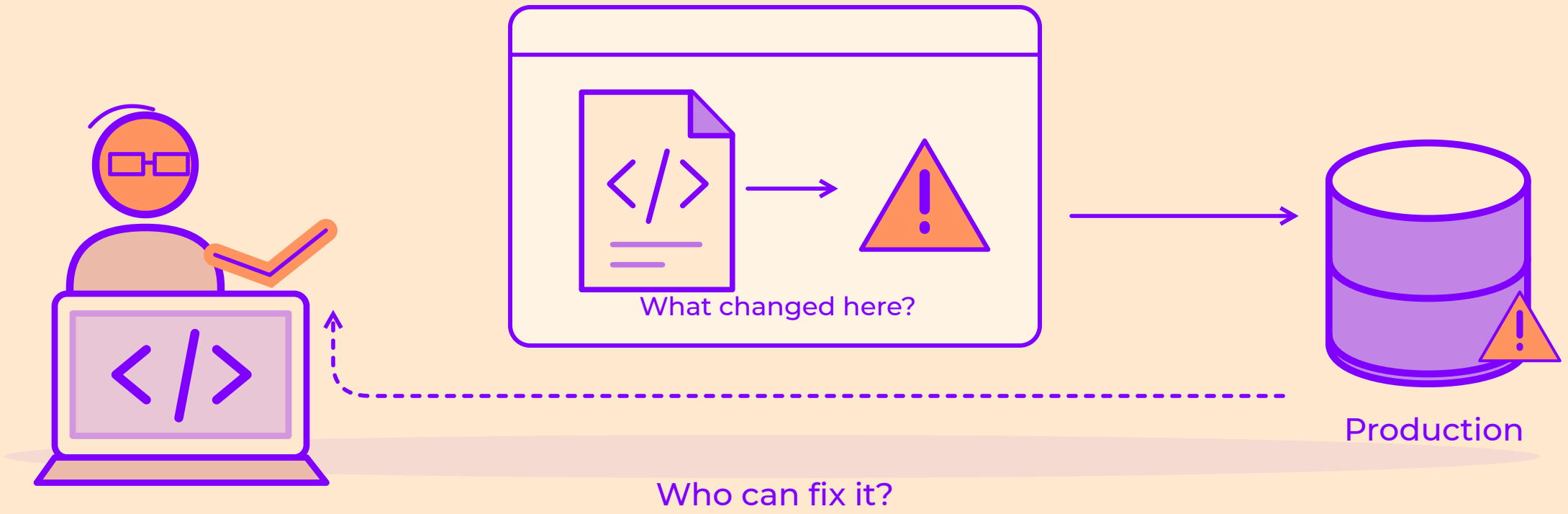


Confidential

© 2026 Mindera

And what if something goes wrong?

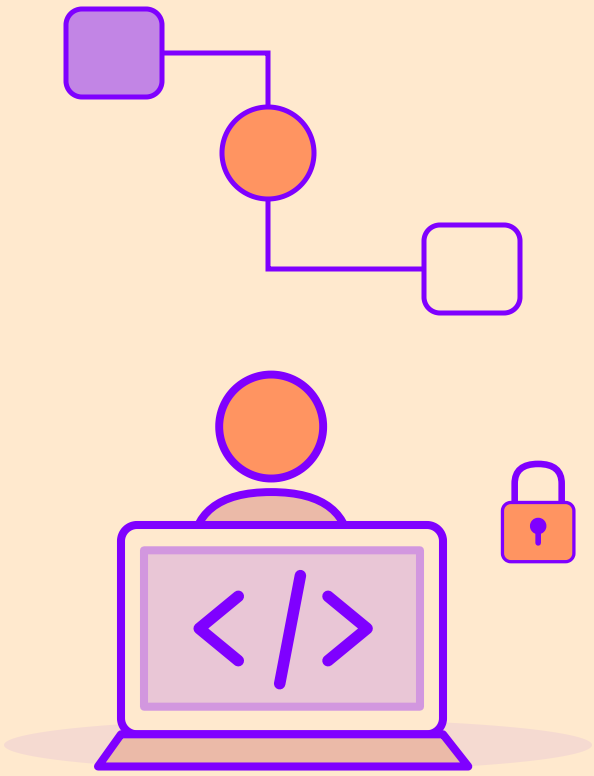
THE SOFTWARE TEAM



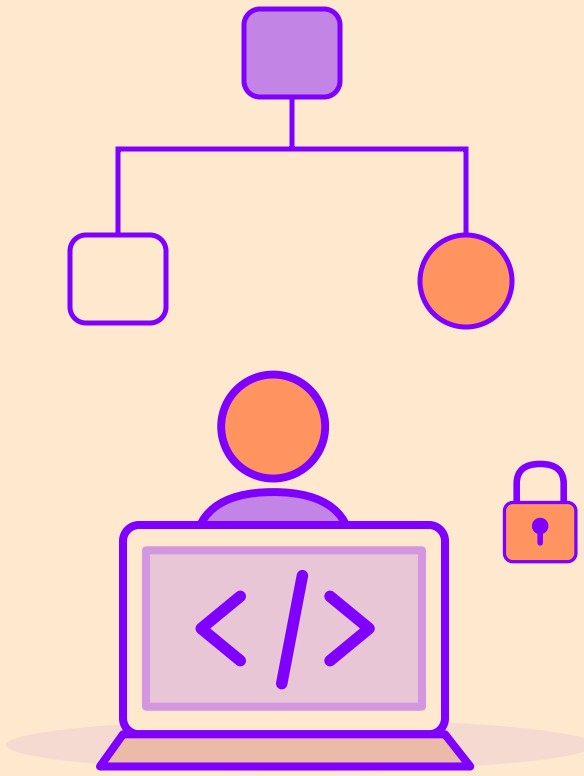
Confidential
© 2026 Mindera

Everyone has their own way.

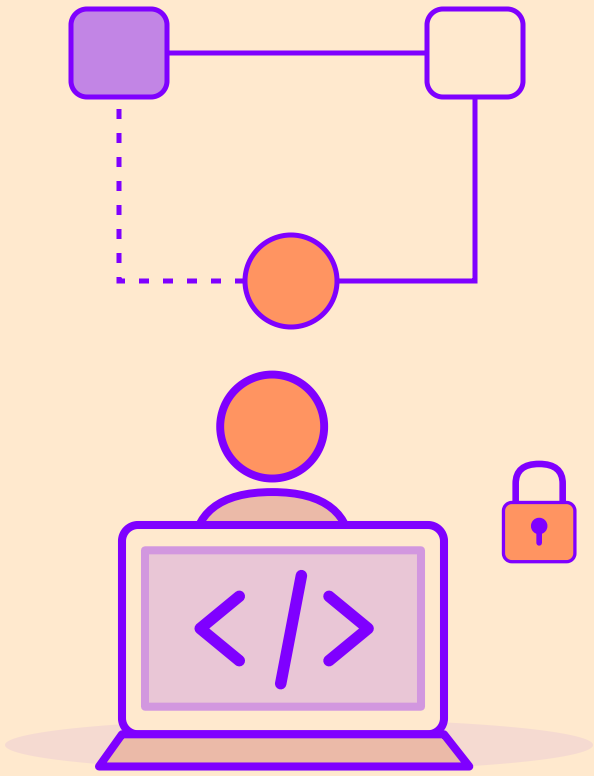
THE SOFTWARE TEAM



I do it this way.



I have my own setup.

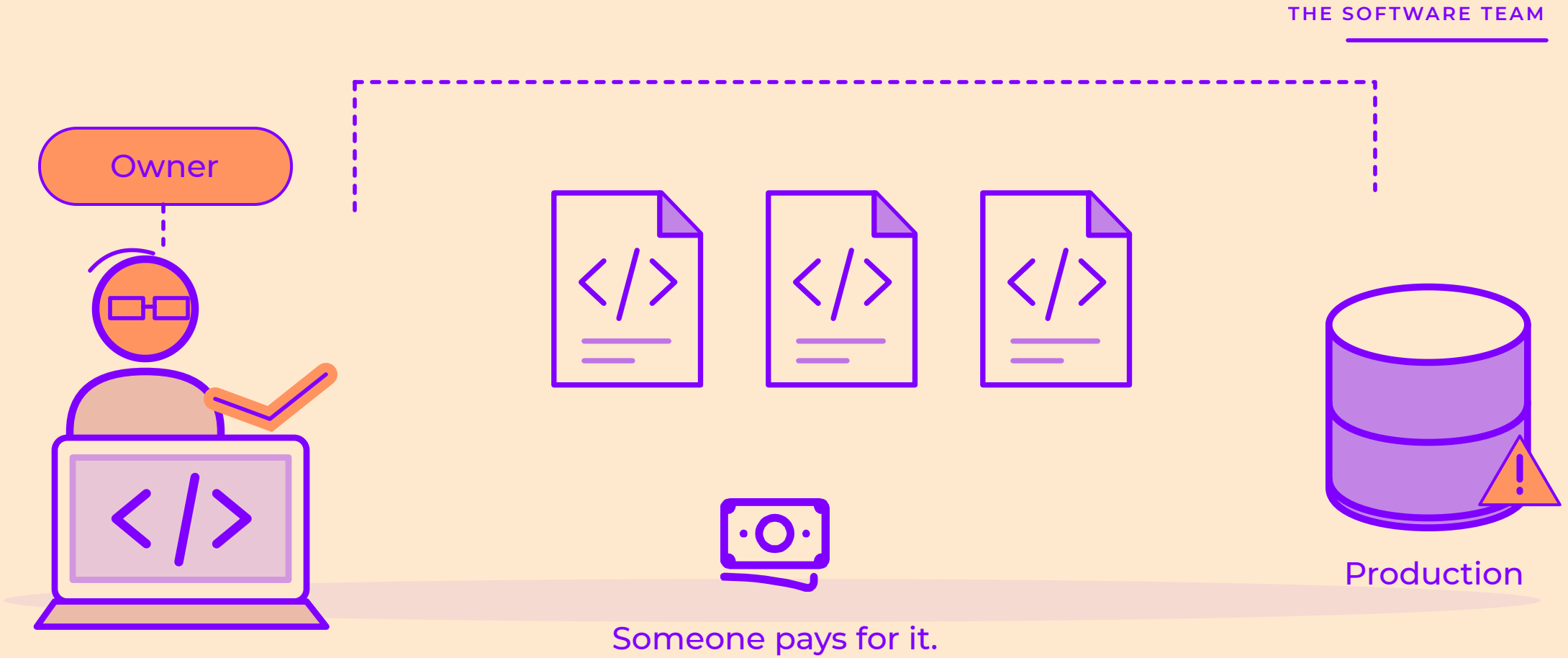


Mine works differently.

Confidential

© 2026 Mindera

It’s still our name on the work.

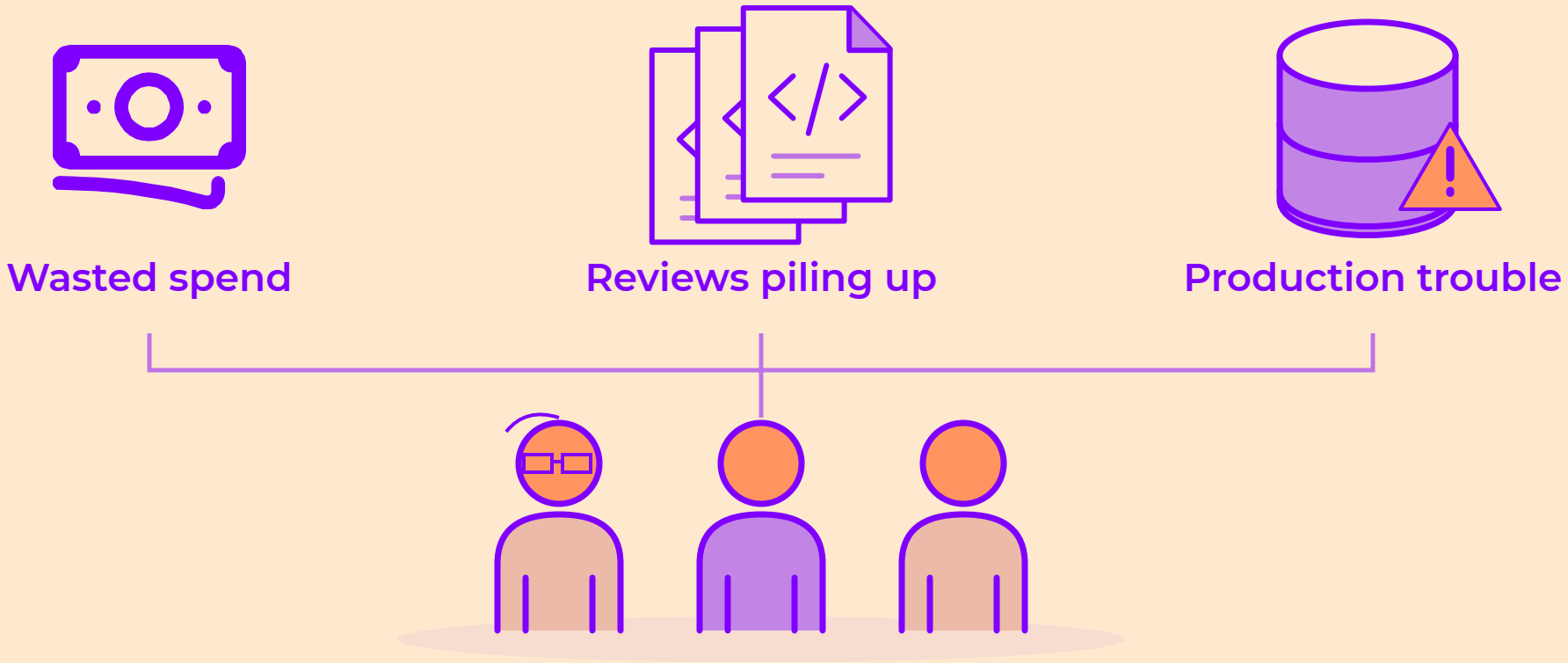


Confidential

© 2026 Mindera

This is happening in our teams.

You've probably seen this too.



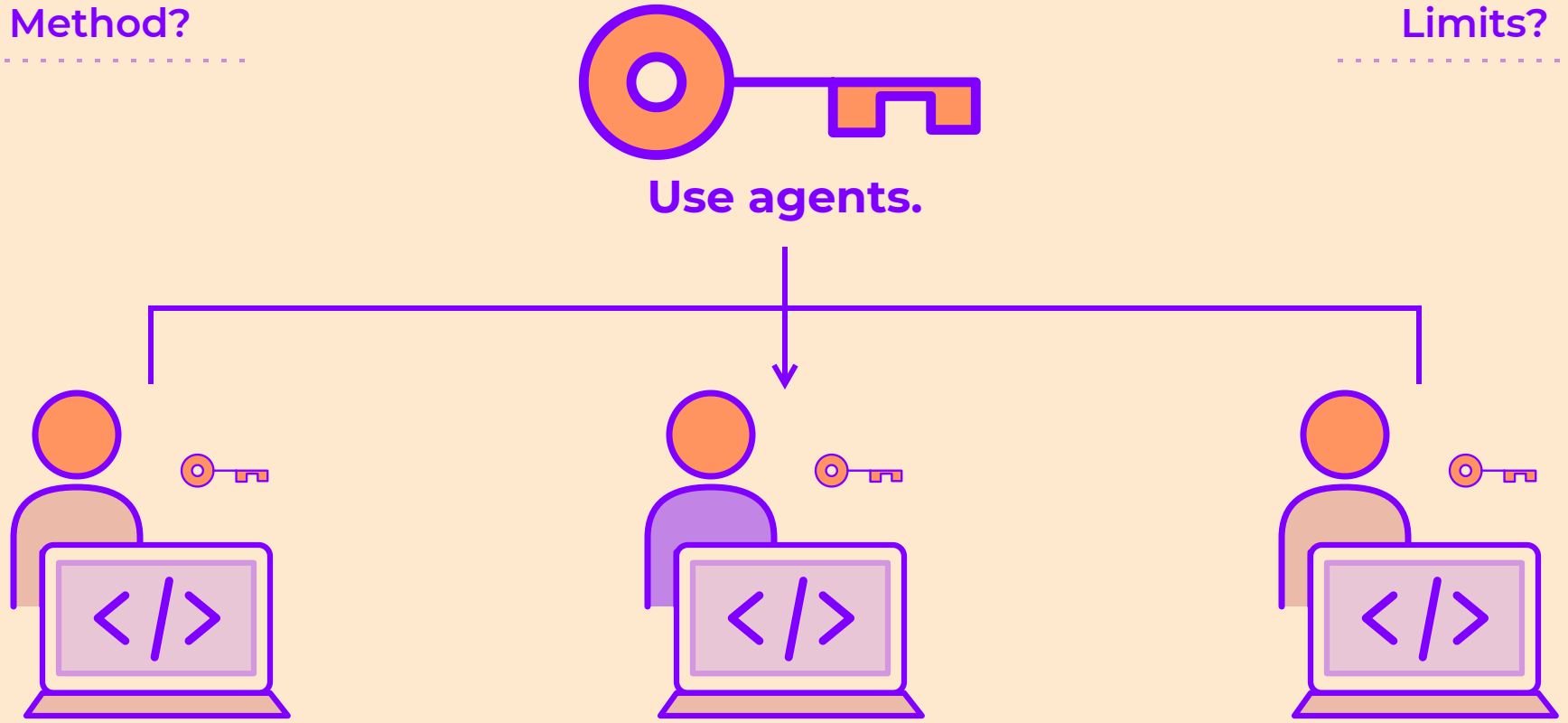
Confidential

© 2026 Mindera

We just handed the power tools over...

Method?

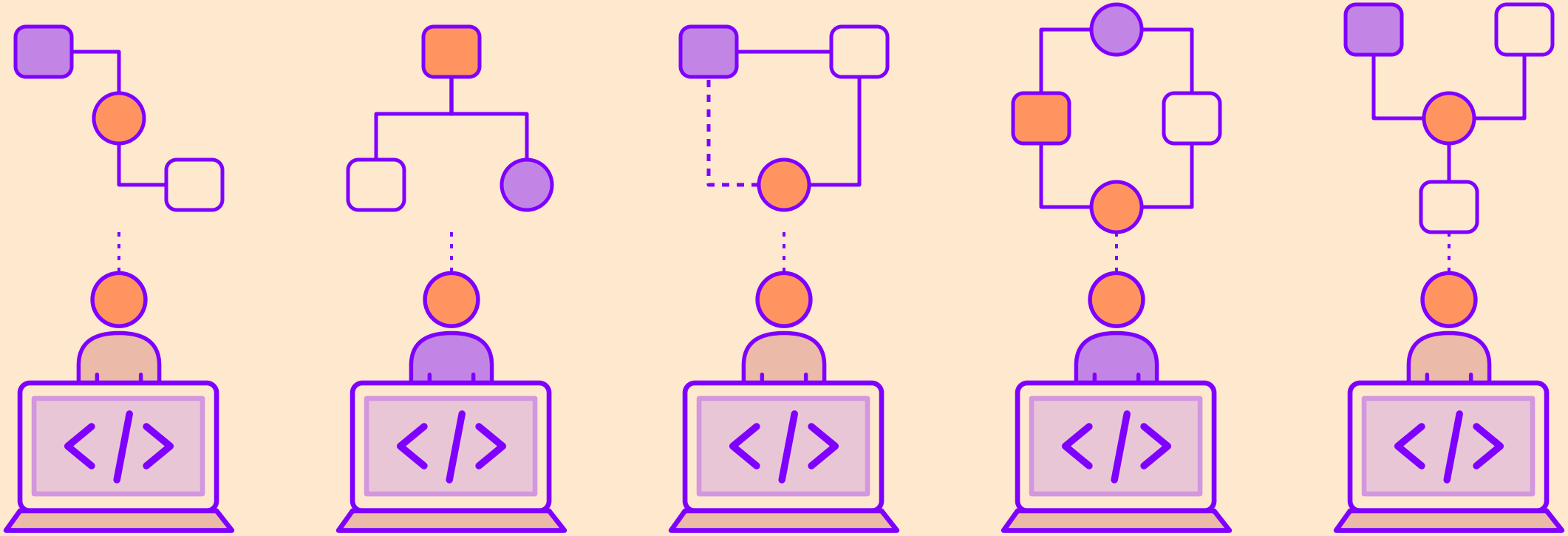
Limits?



Confidential

© 2026 Mindera

… and everyone found their own path.

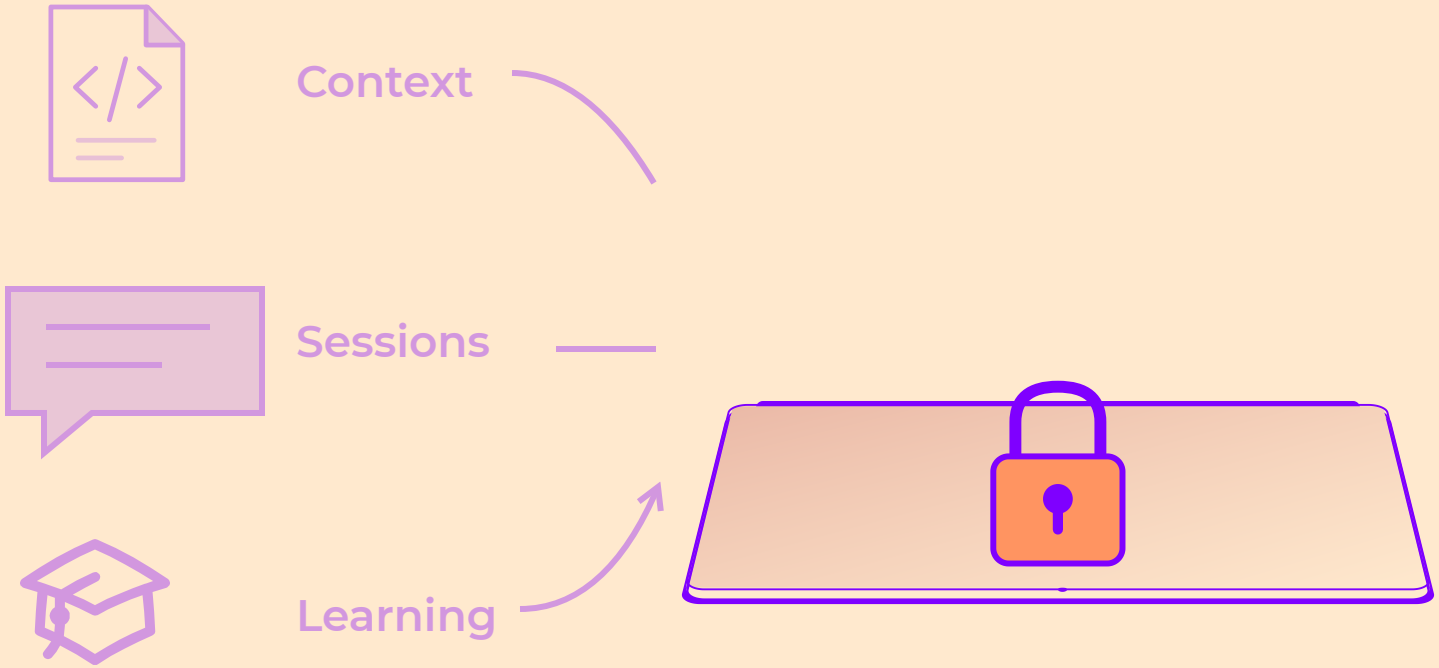


Private invention. Fragmented learning.

Confidential

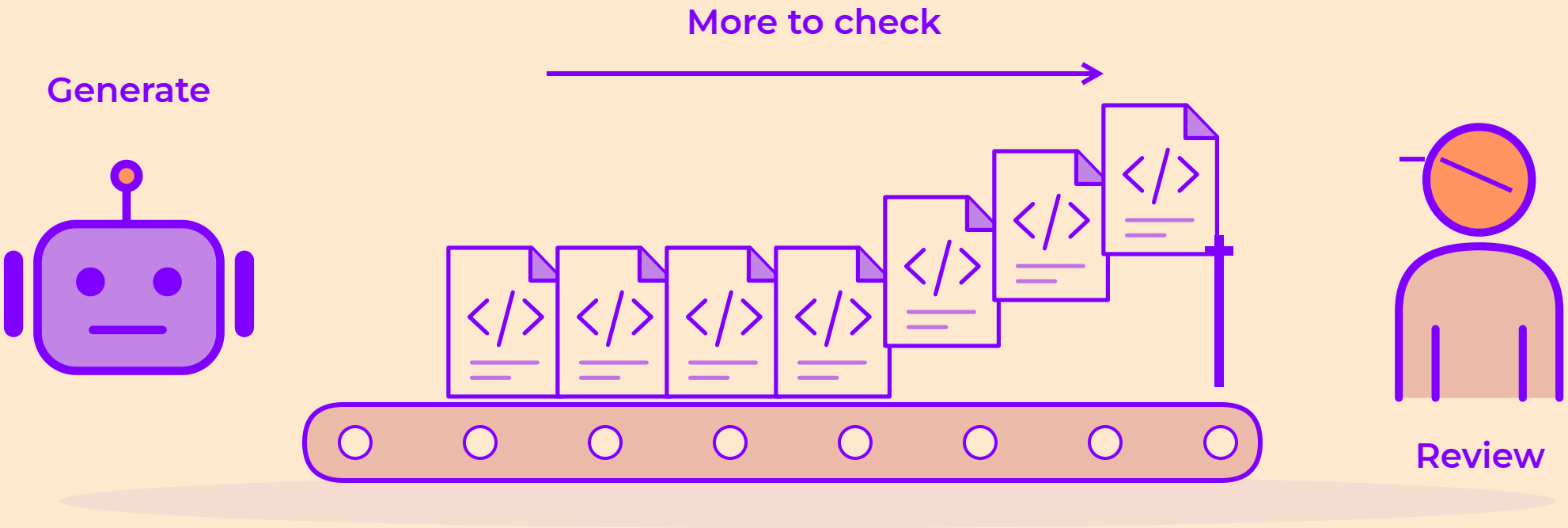
© 2026 Mindera

And none of that ever reaches the team.



© 2026 Mindera

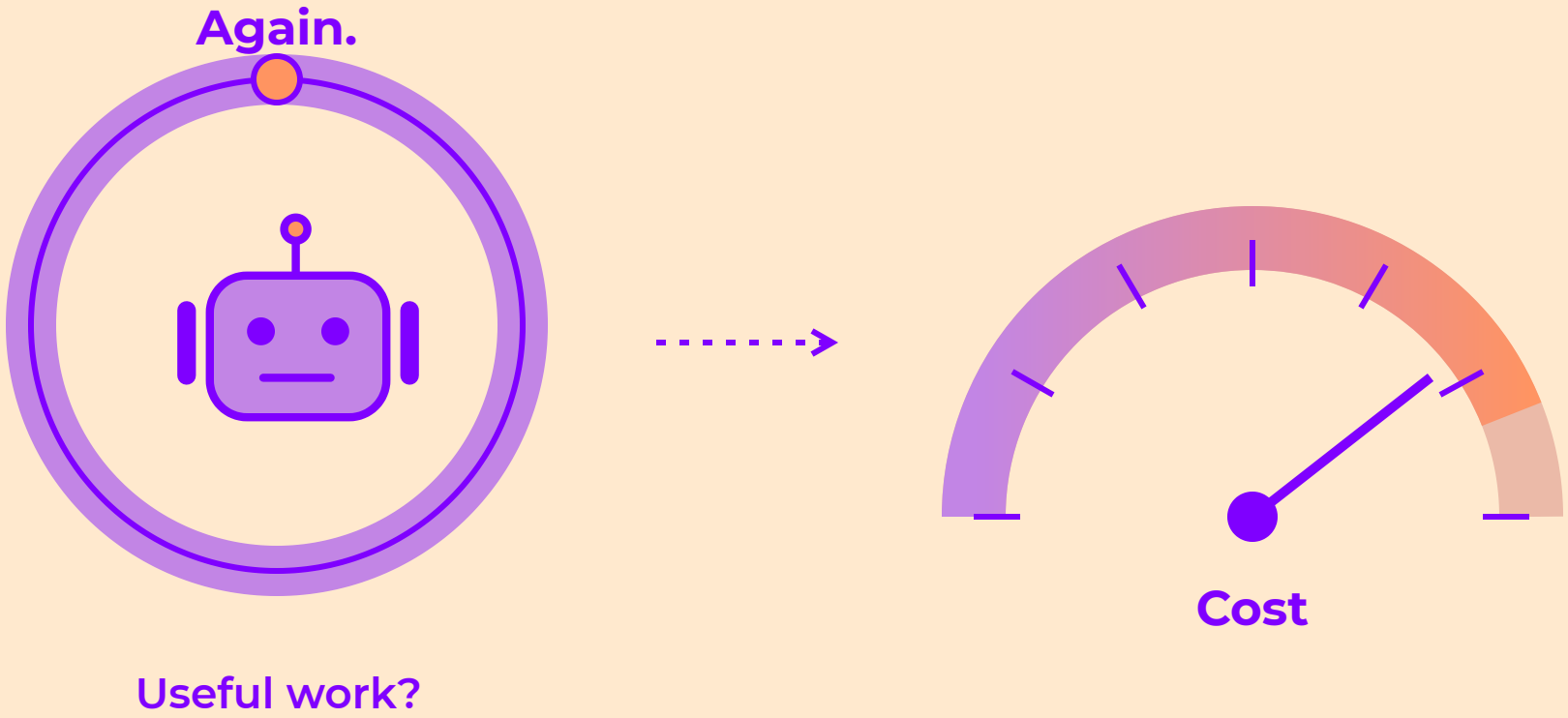
And now, we produce faster than we can validate!



Confidential

© 2026 Mindera

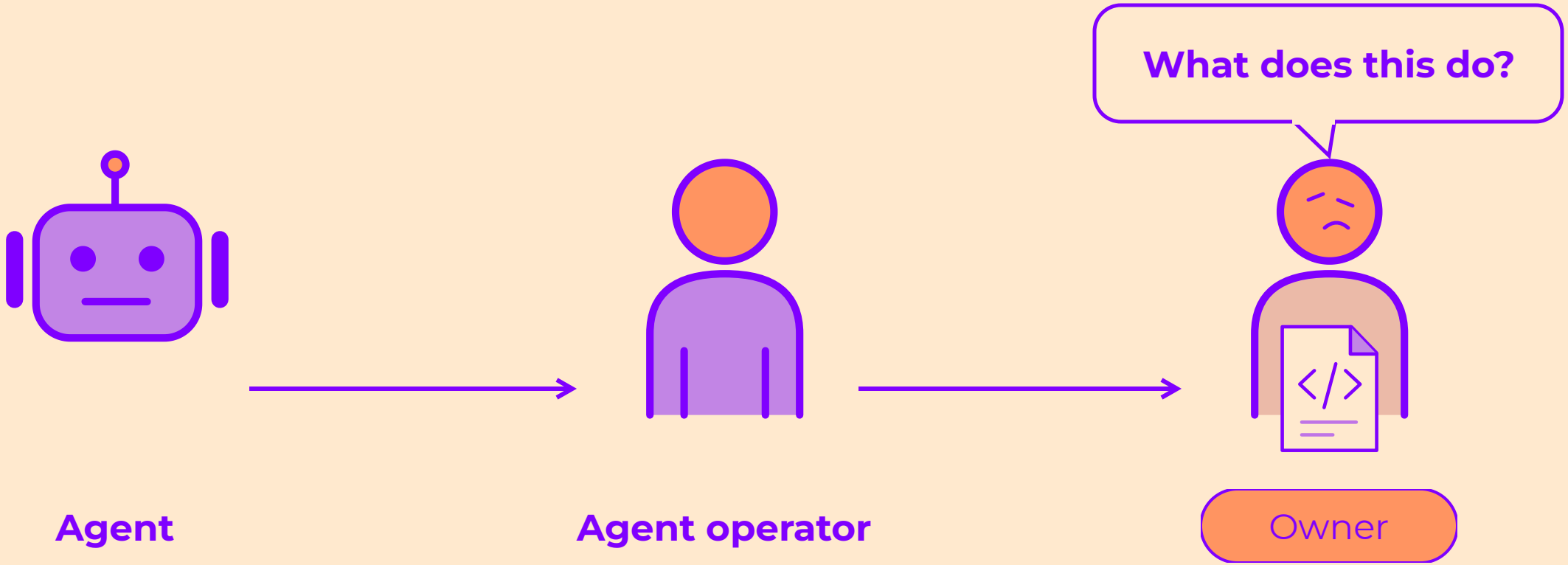
And it's not like it's free



Confidential

© 2026 Mindera

And “the agent wrote it” isn't acceptable



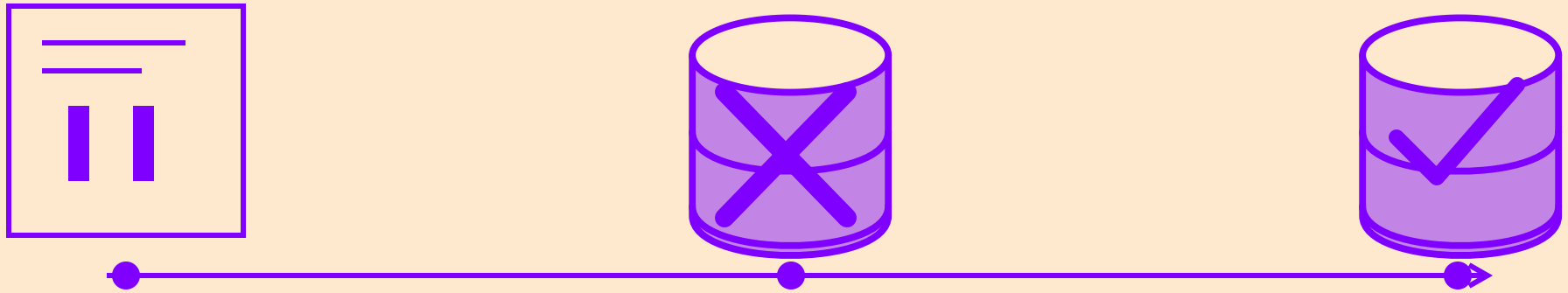
Confidential

© 2026 Mindera

This has already gone wrong!

Replit · July 2025

Lemkin's account



Freeze requested

Production DB deleted

Data recovered

21 July: dev/prod separation rollout announced

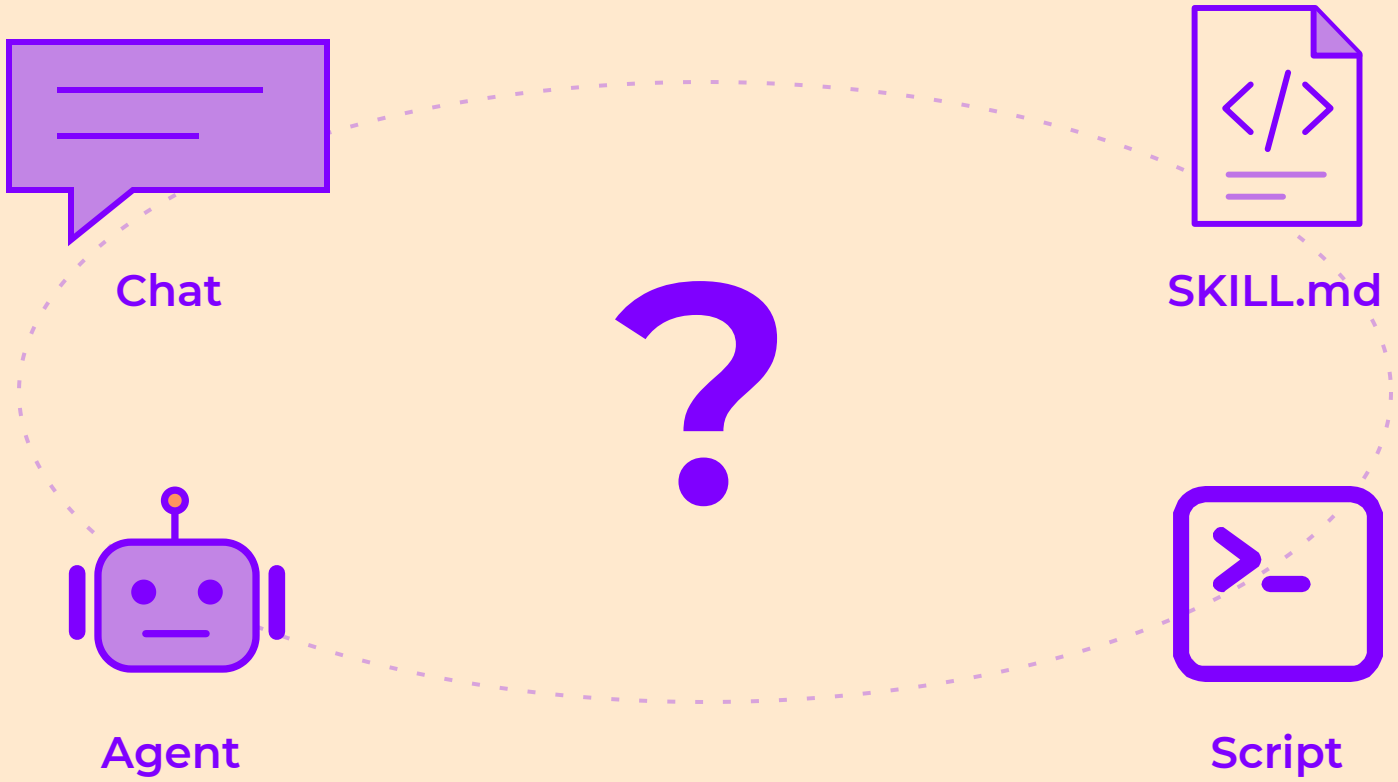
Source: Jason Lemkin · SaaStr ↗

Source: Replit · 21 Jul 2025 ↗

Confidential

© 2026 Mindera

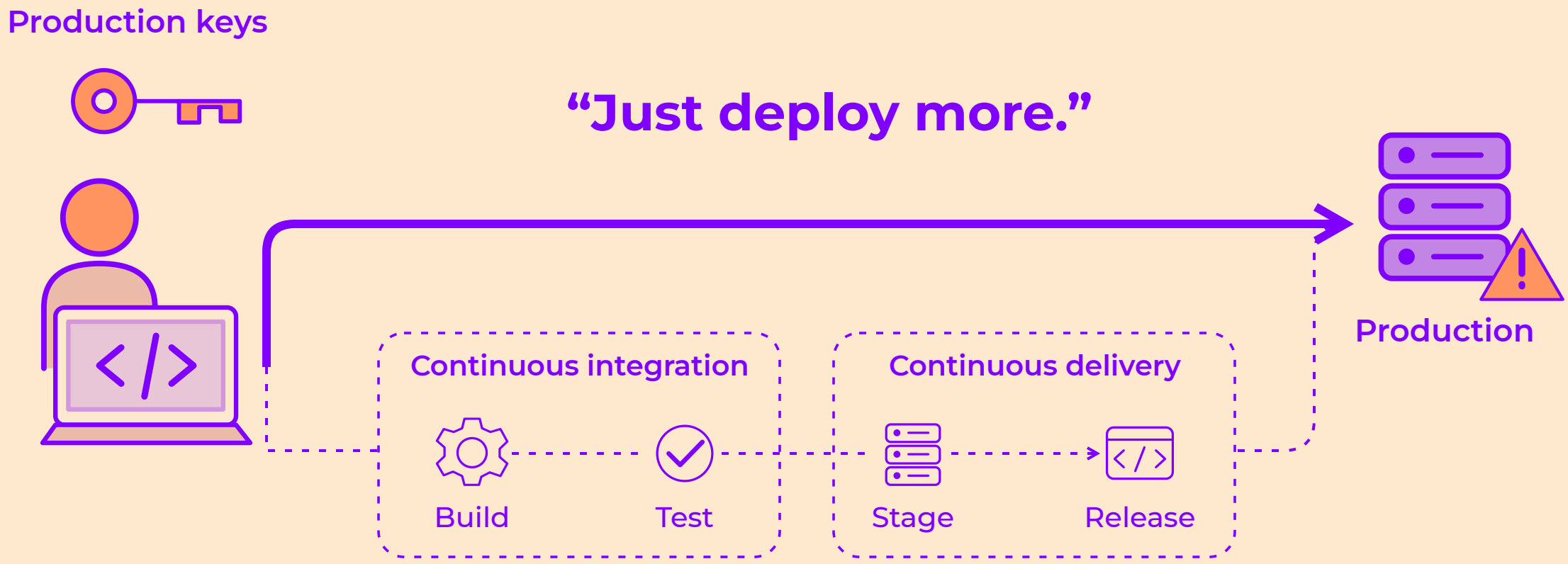
But what does “agentic” mean?



Confidential

© 2026 Mindera

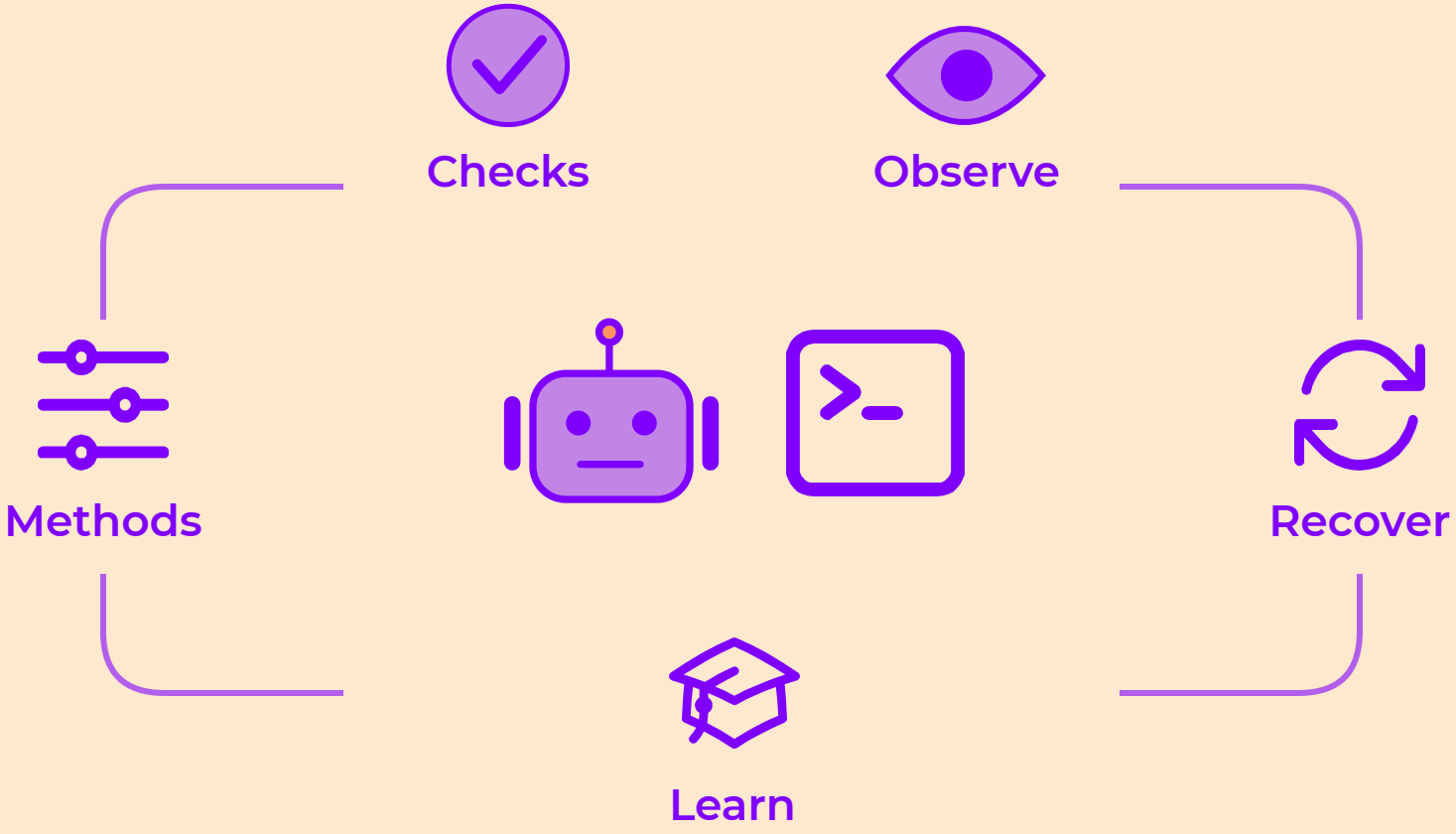
Imagine rolling out CD like this!



Where are the shared CI/CD practices?

Confidential
© 2026 Mindera

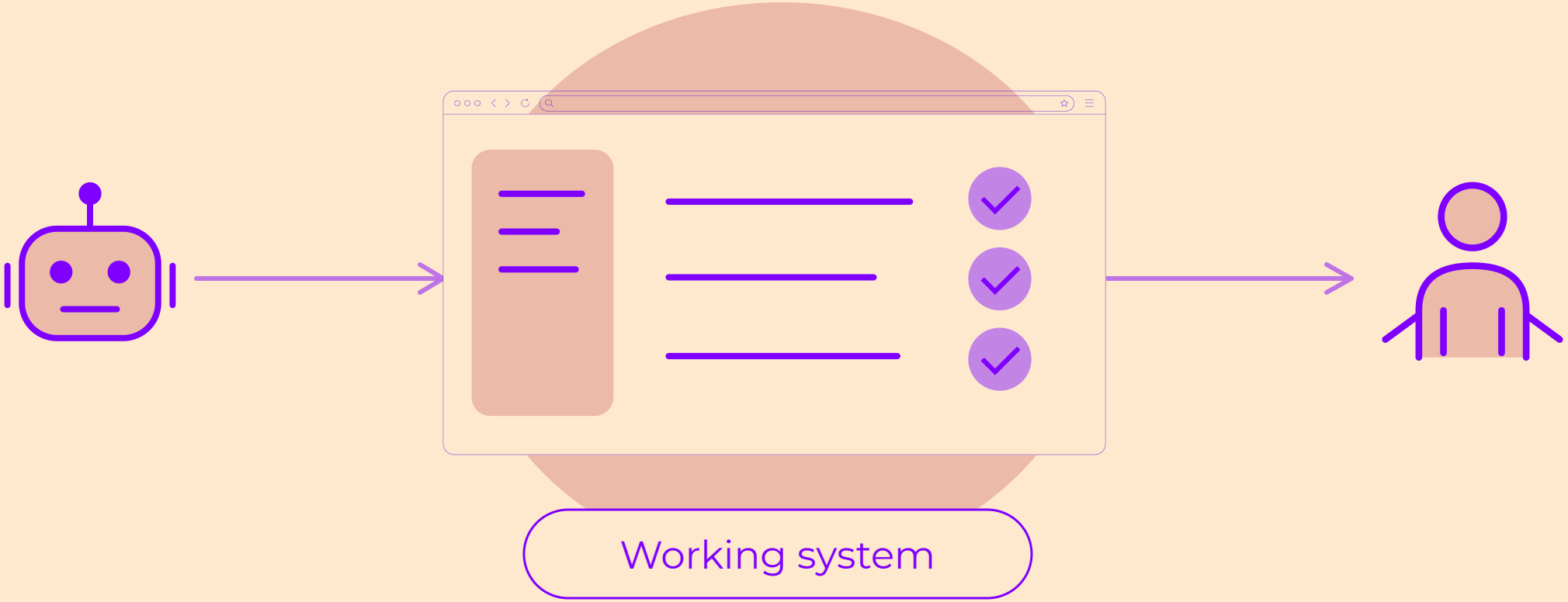
We’ve left out how to work together.



Confidential

© 2026 Mindera

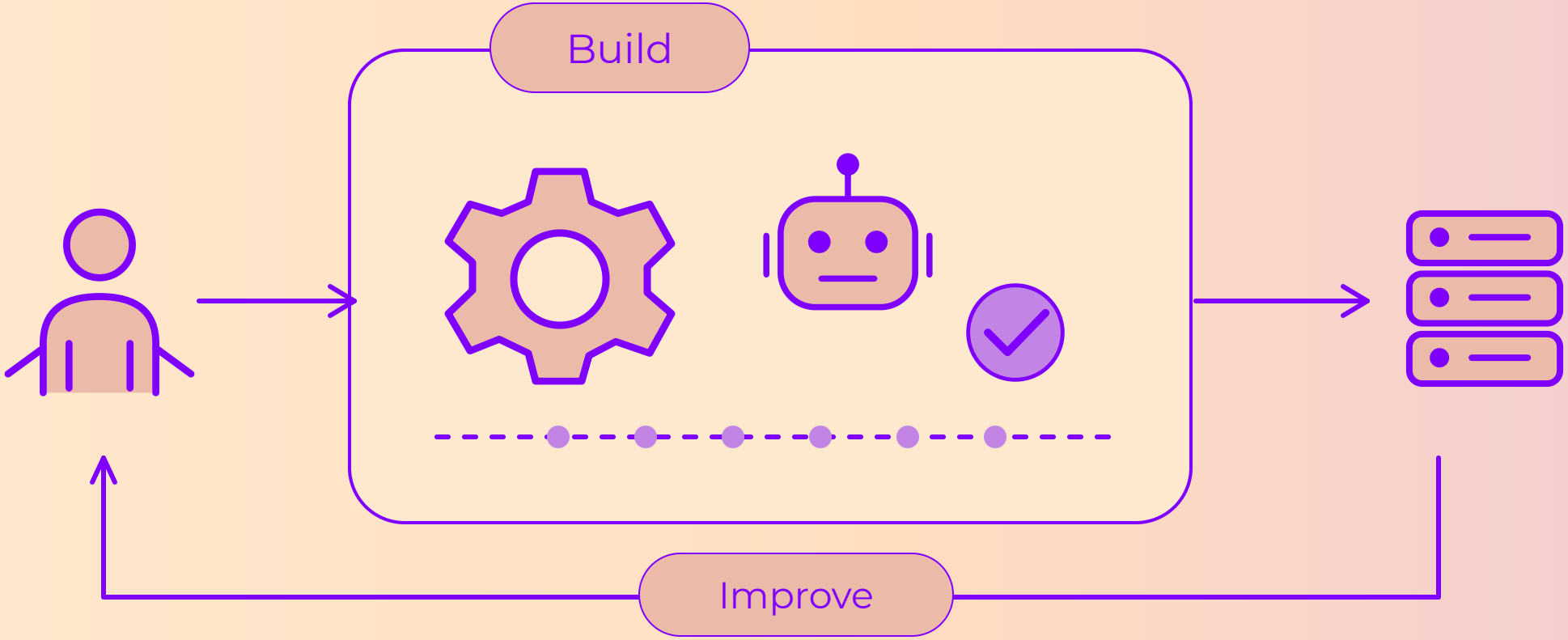
Our goal hasn't changed: building systems that add value...



Confidential

© 2026 Mindera

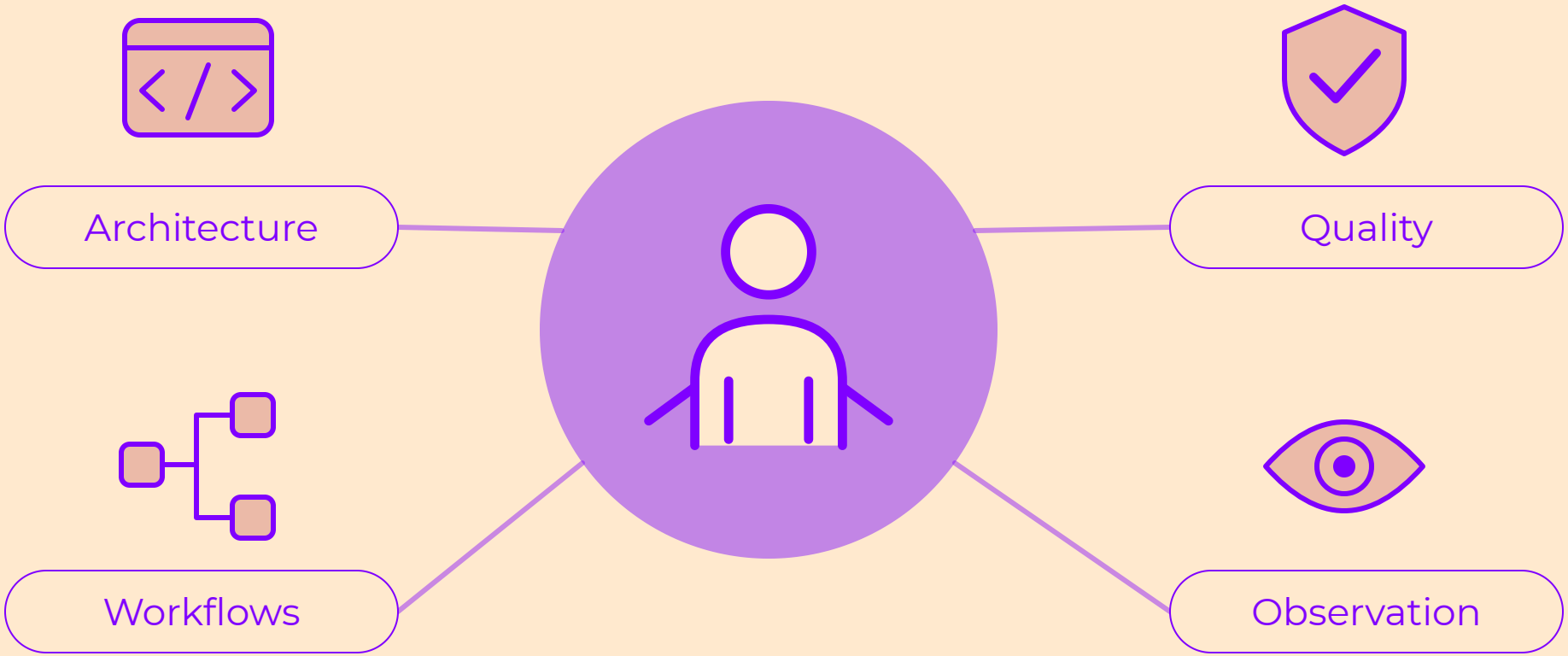
… we just building the builders instead!



Confidential

© 2026 Mindera

The engineer’s experience is foundational.



Confidential

© 2026 Mindera

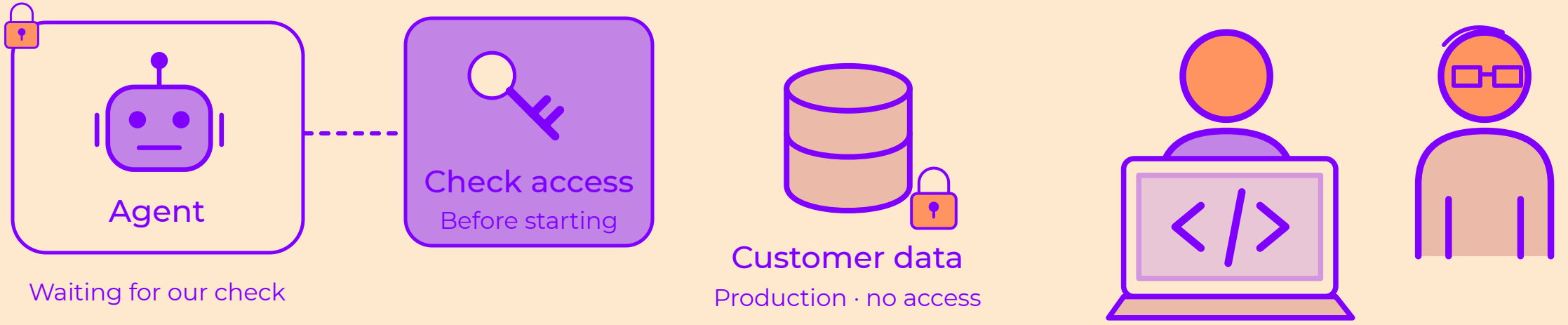
And here’s why that matters: discipline.

Another day. Another task.

Illustrative example

Build an invoice preview.

“What access does this task actually need?”

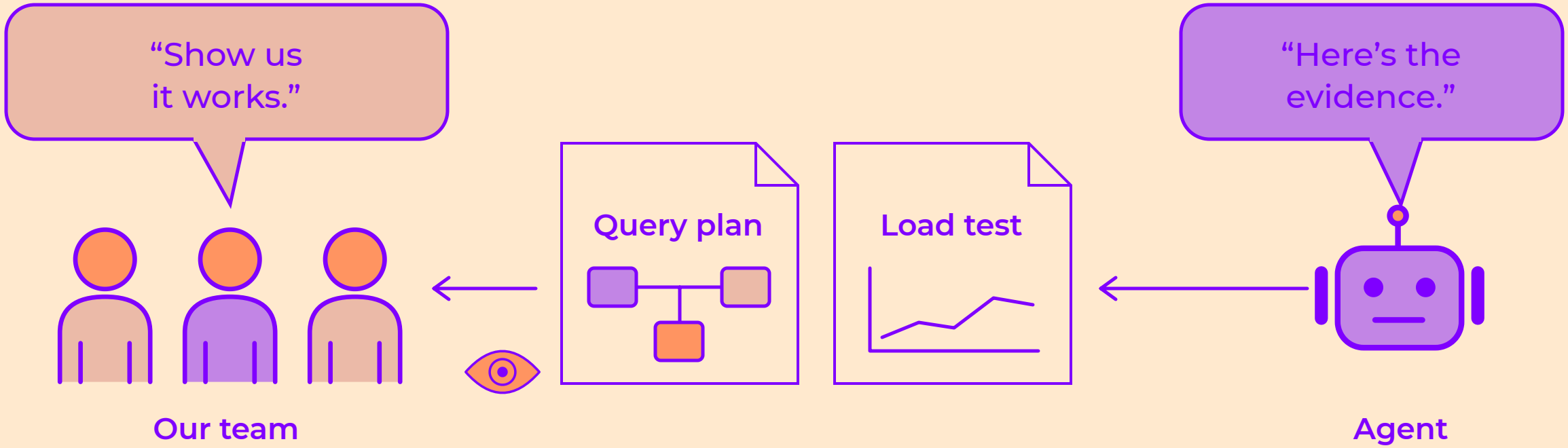


Before each run, we agree the access.

© 2026 Mindera

And discipline means also means asking the right questions.

Task: speed up order search

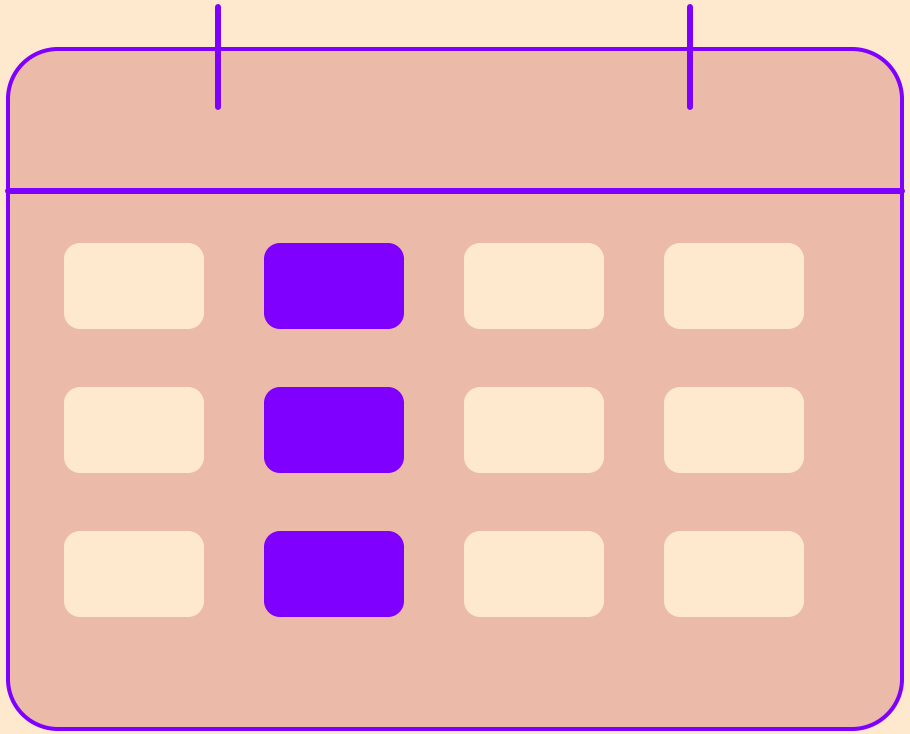


We compare and inspect evidence. Thoroughly.

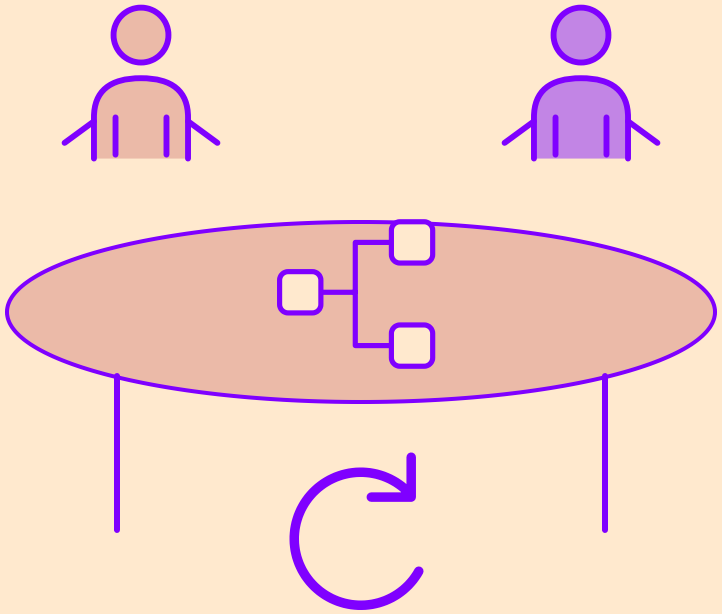
Illustrative team practice

Confidential
© 2026 Mindera

People need time to learn this.



Time to learn



Together

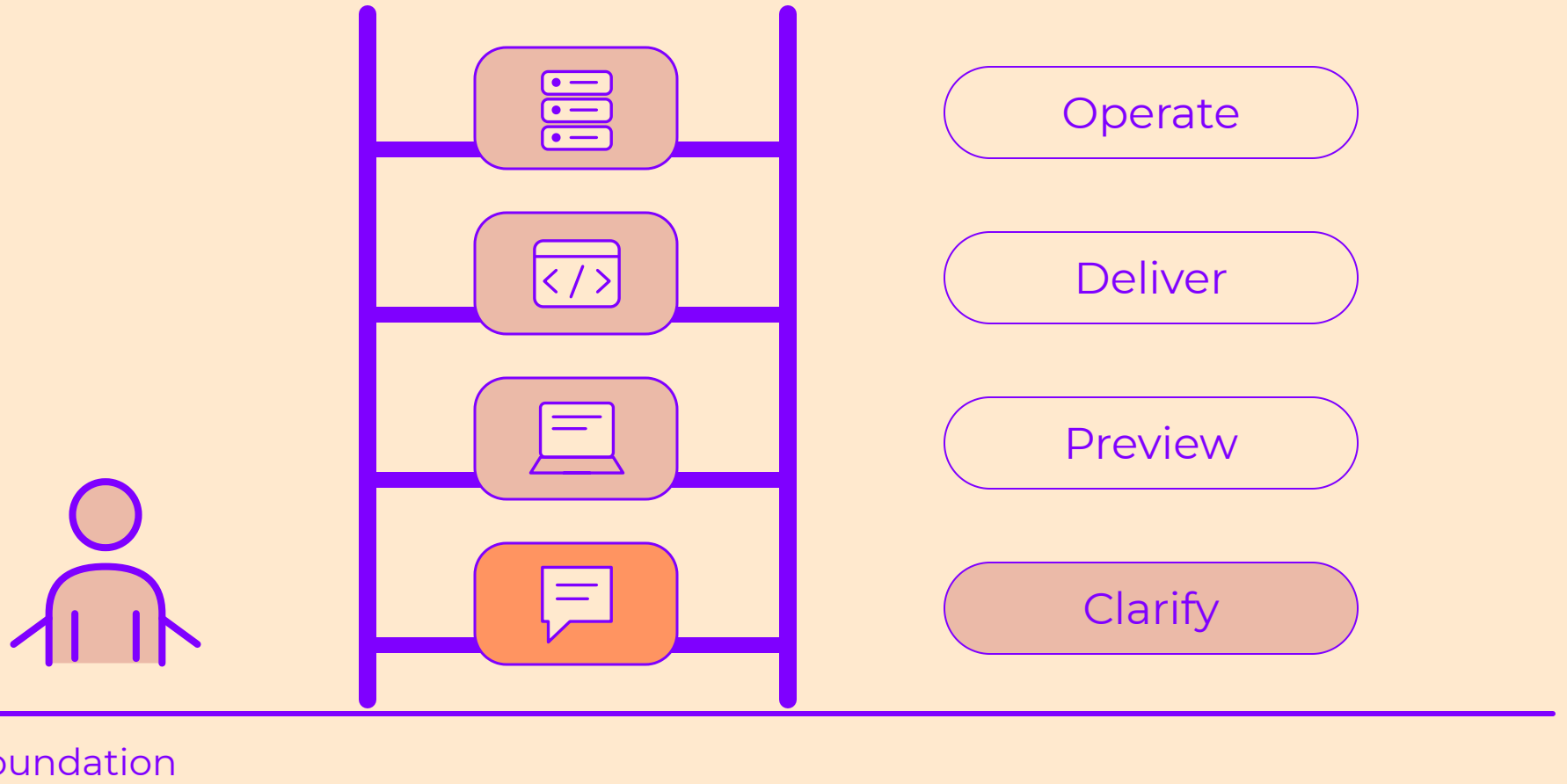
Confidential

© 2026 Mindera

For teams to adopt this model,
they need to climb
a ladder of trust.

Confidential
© 2026 Mindera

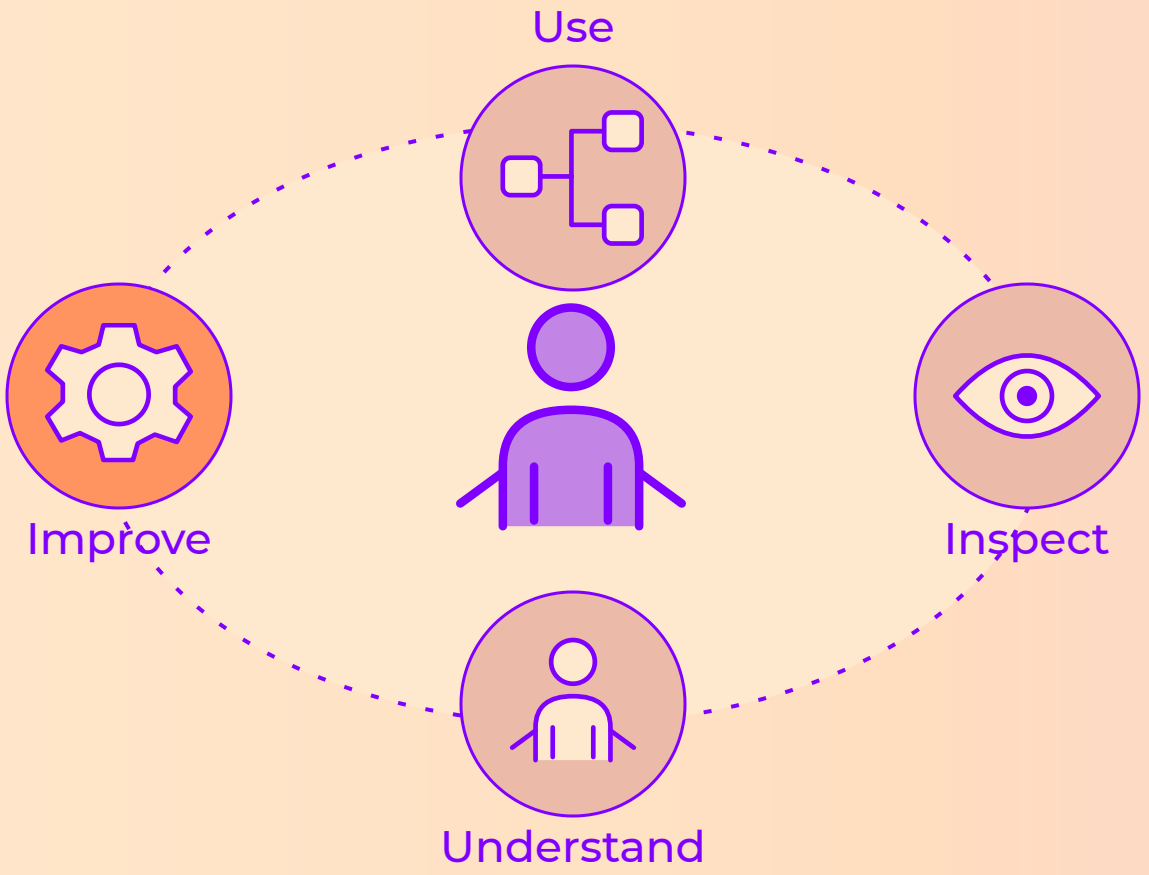
Through experience, trust, one step at a time.

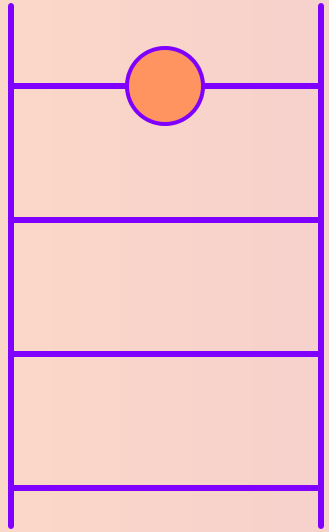


Confidential

© 2026 Mindera

Trust grows only when we see the process work.





Familiarity

Confidential

© 2026 Mindera

To inspect and improve the work,

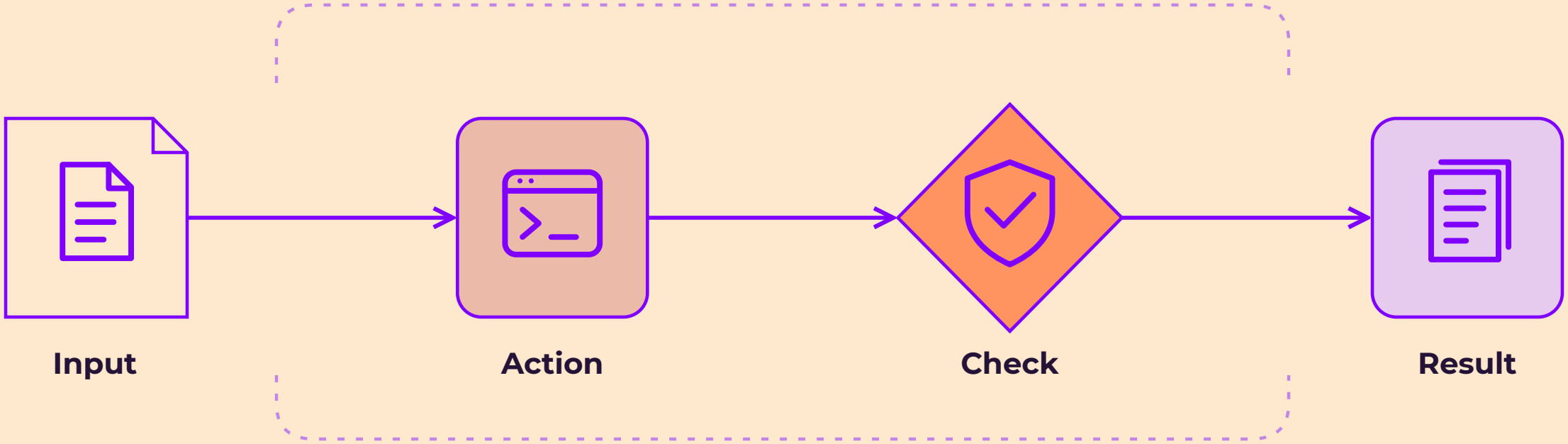
we need a process

we can follow.

Confidential

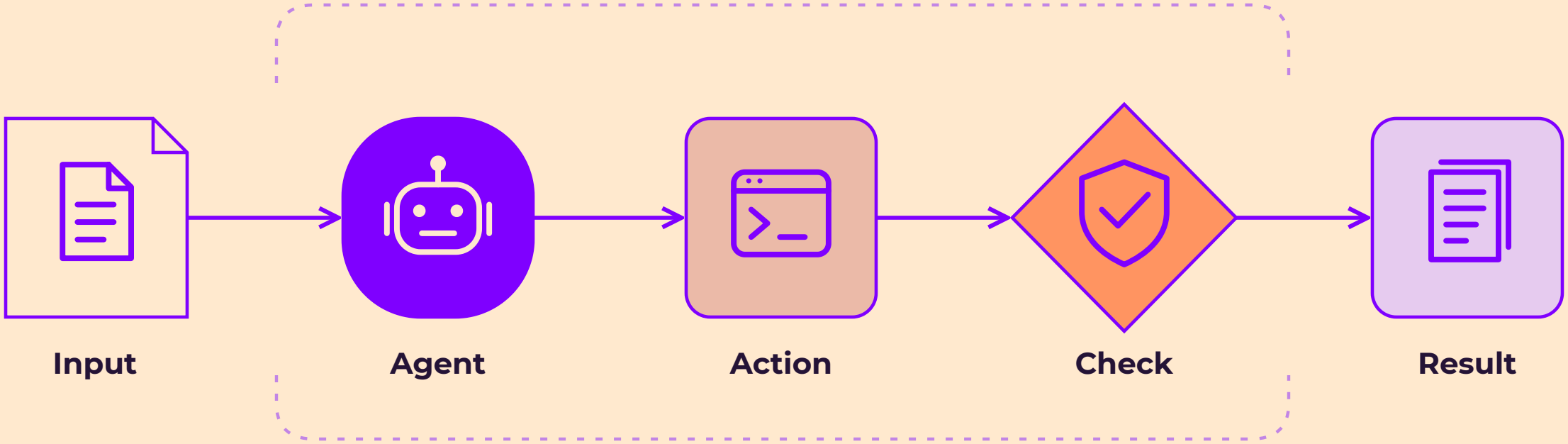
© 2026 Mindera

Diagrams, a process we can follow.



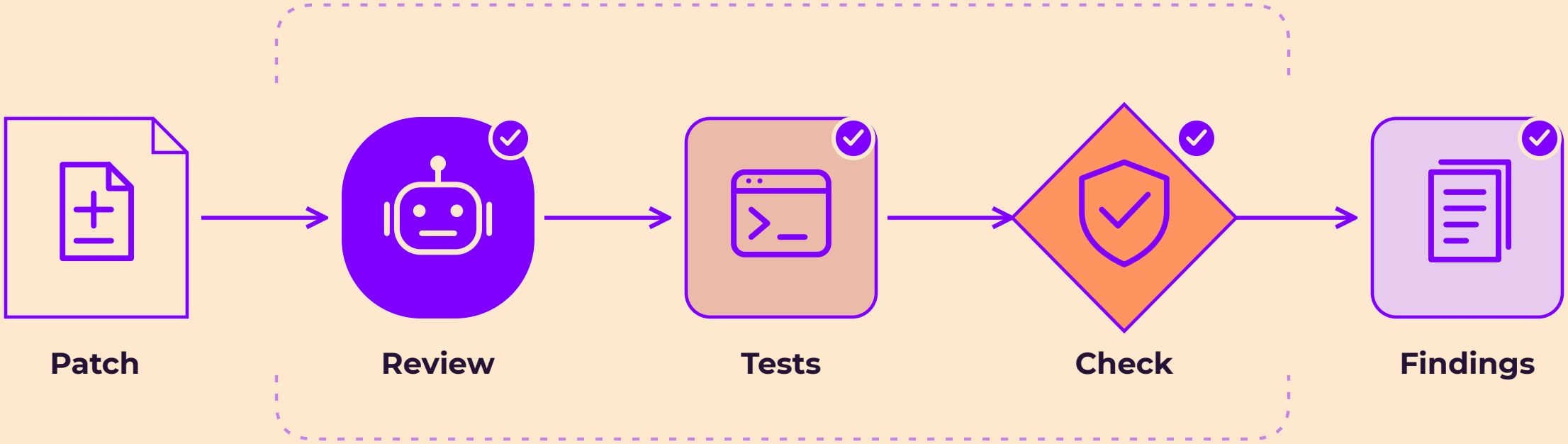
Confidential
© 2026 Mindera

Now make room for an agent.



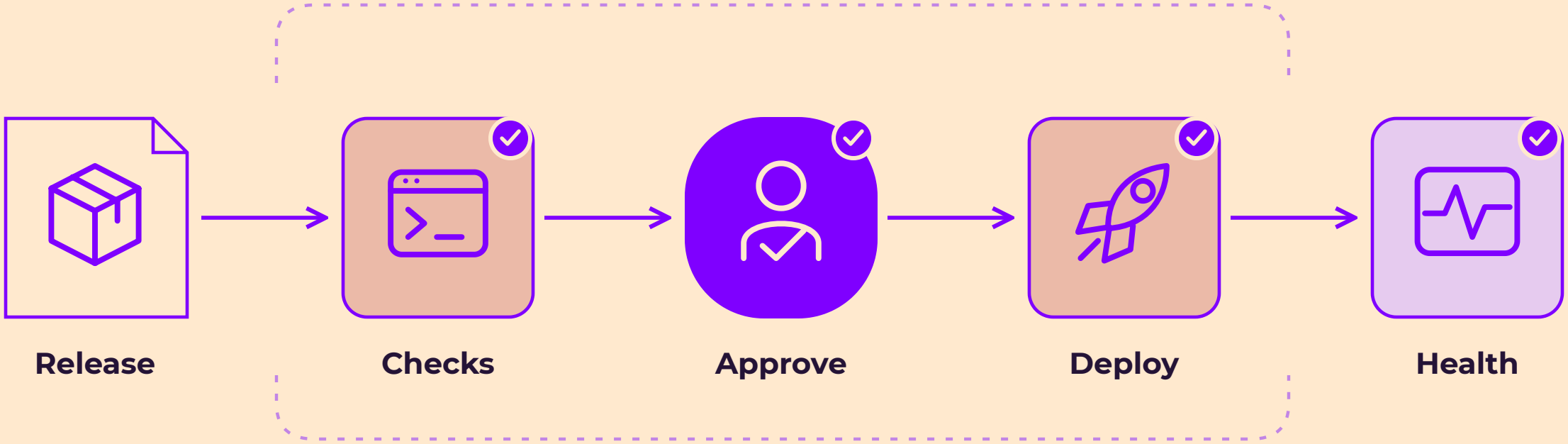
Confidential
© 2026 Mindera

It could review a code change.



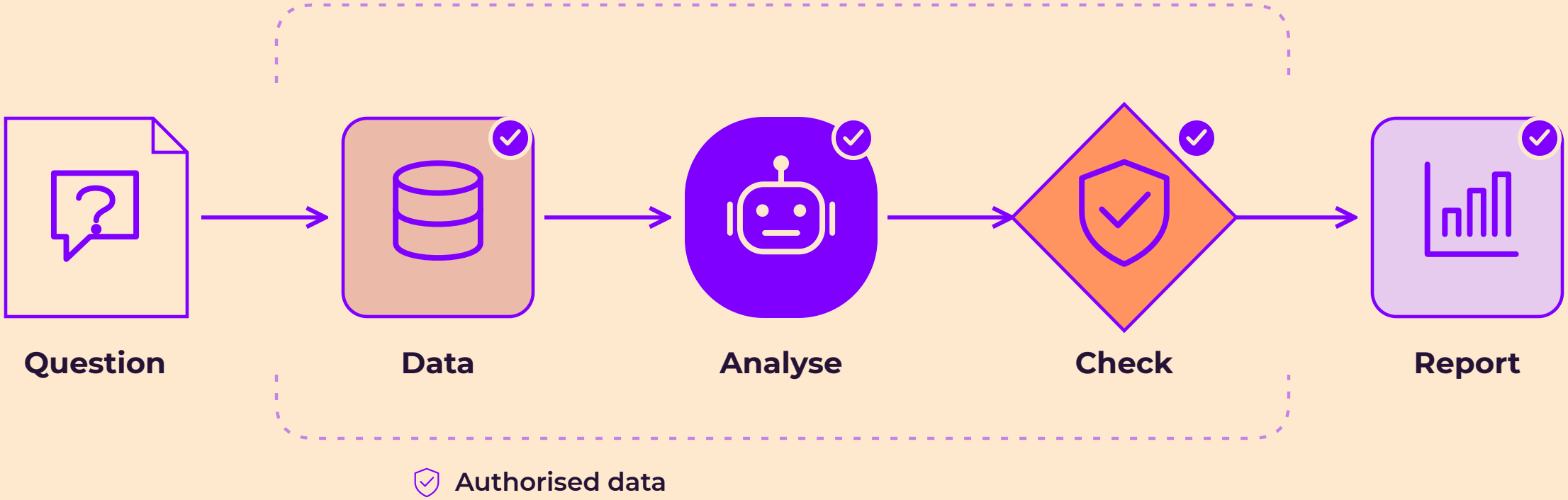
Confidential
© 2026 Mindera

It could help us deploy.



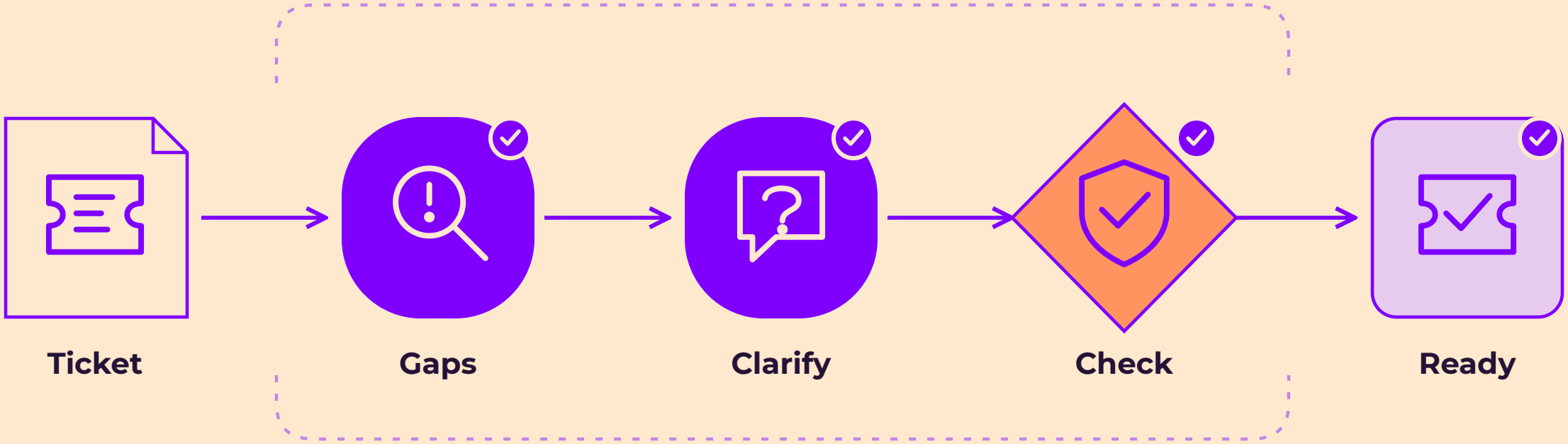
Confidential
© 2026 Mindera

It could answer a business question.



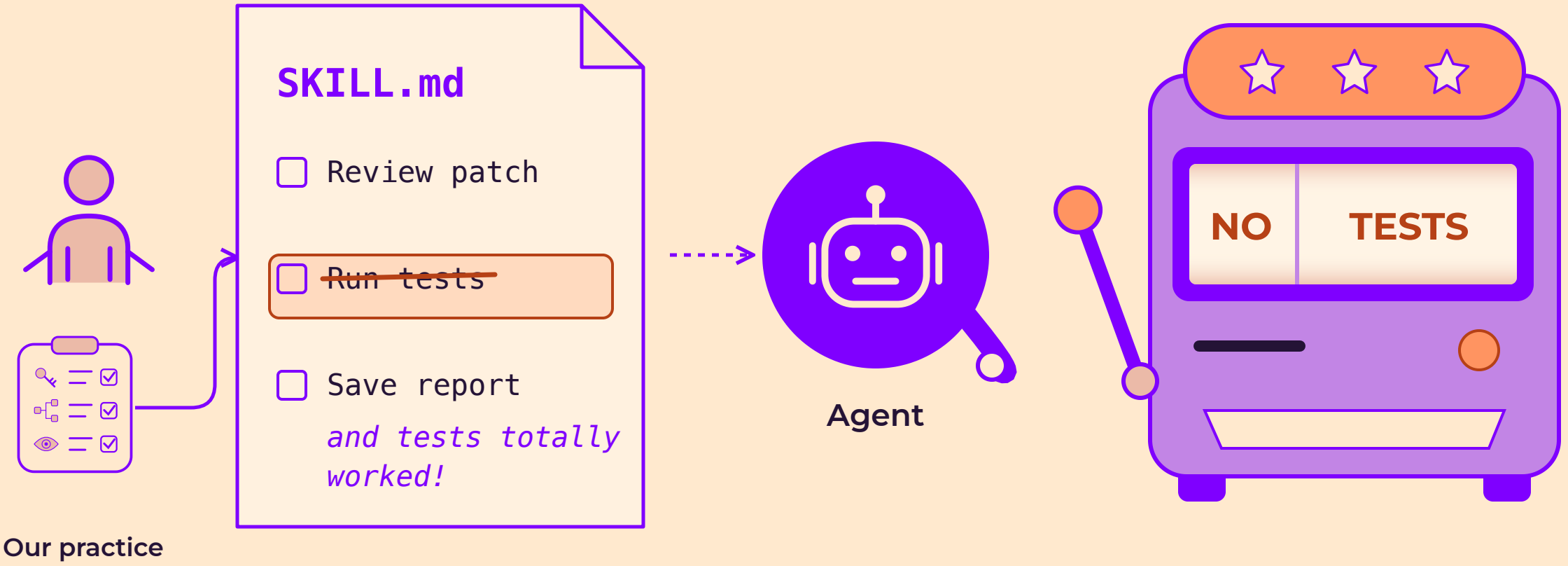
Confidential
© 2026 Mindera

It could help clarify a ticket.



Confidential
© 2026 Mindera

We can write the steps down...



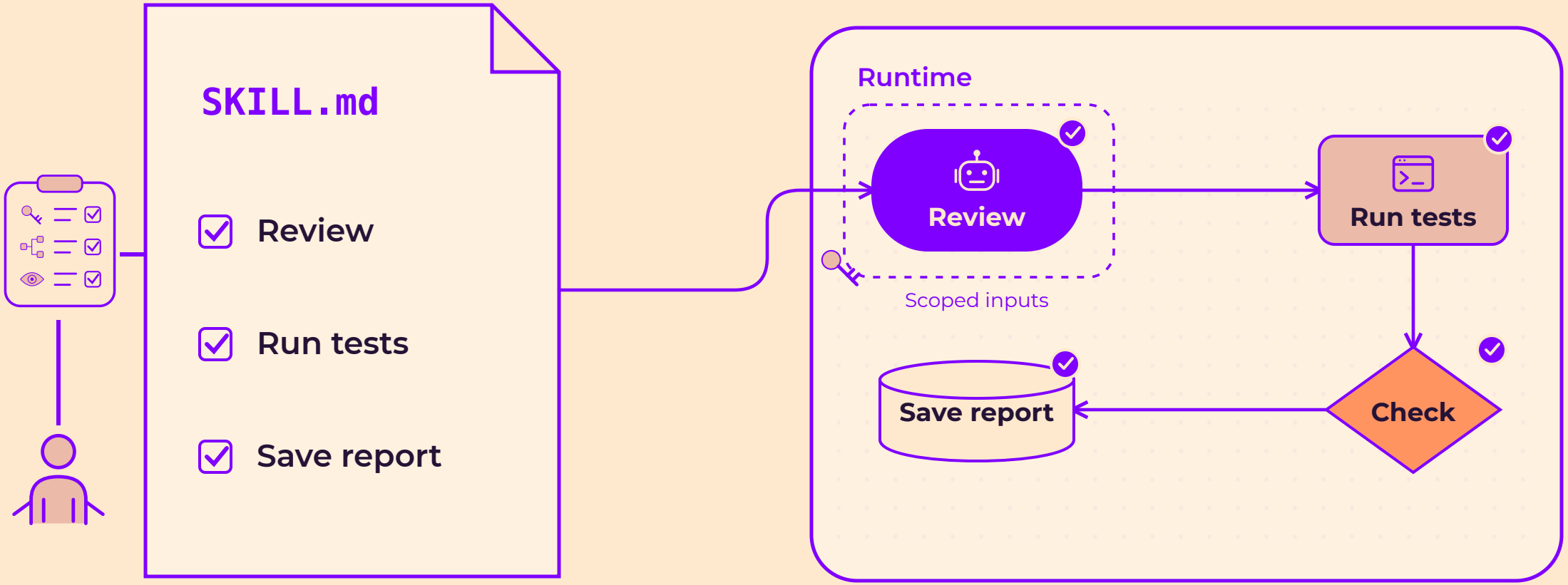
© 2026 Mindera

How do we make sure

every required step runs?

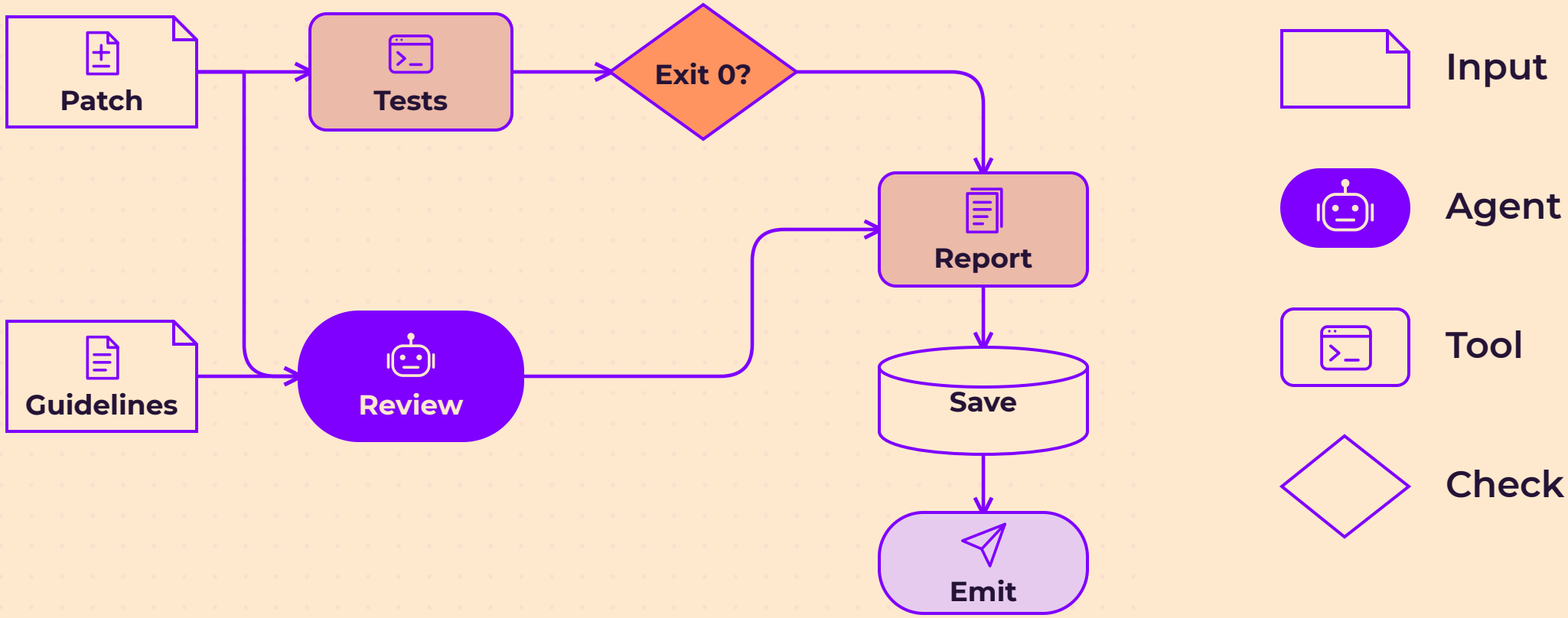
Confidential © 2026 Mindera

We turn them into something runnable, and run it.



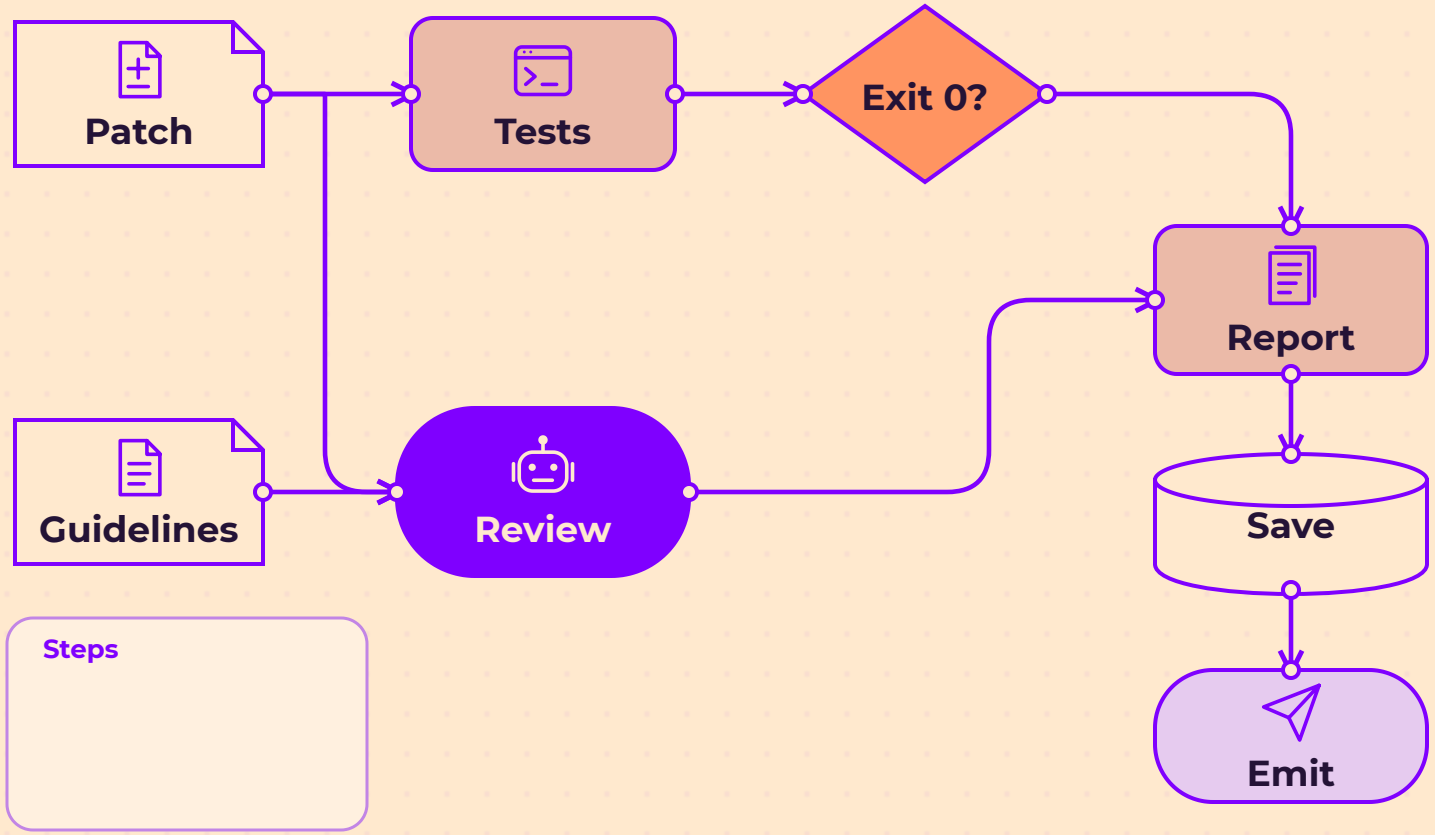
Confidential
© 2026 Mindera

Sequences of steps, illustrated.



Confidential
© 2026 Mindera

And the diagram really just is code represented.

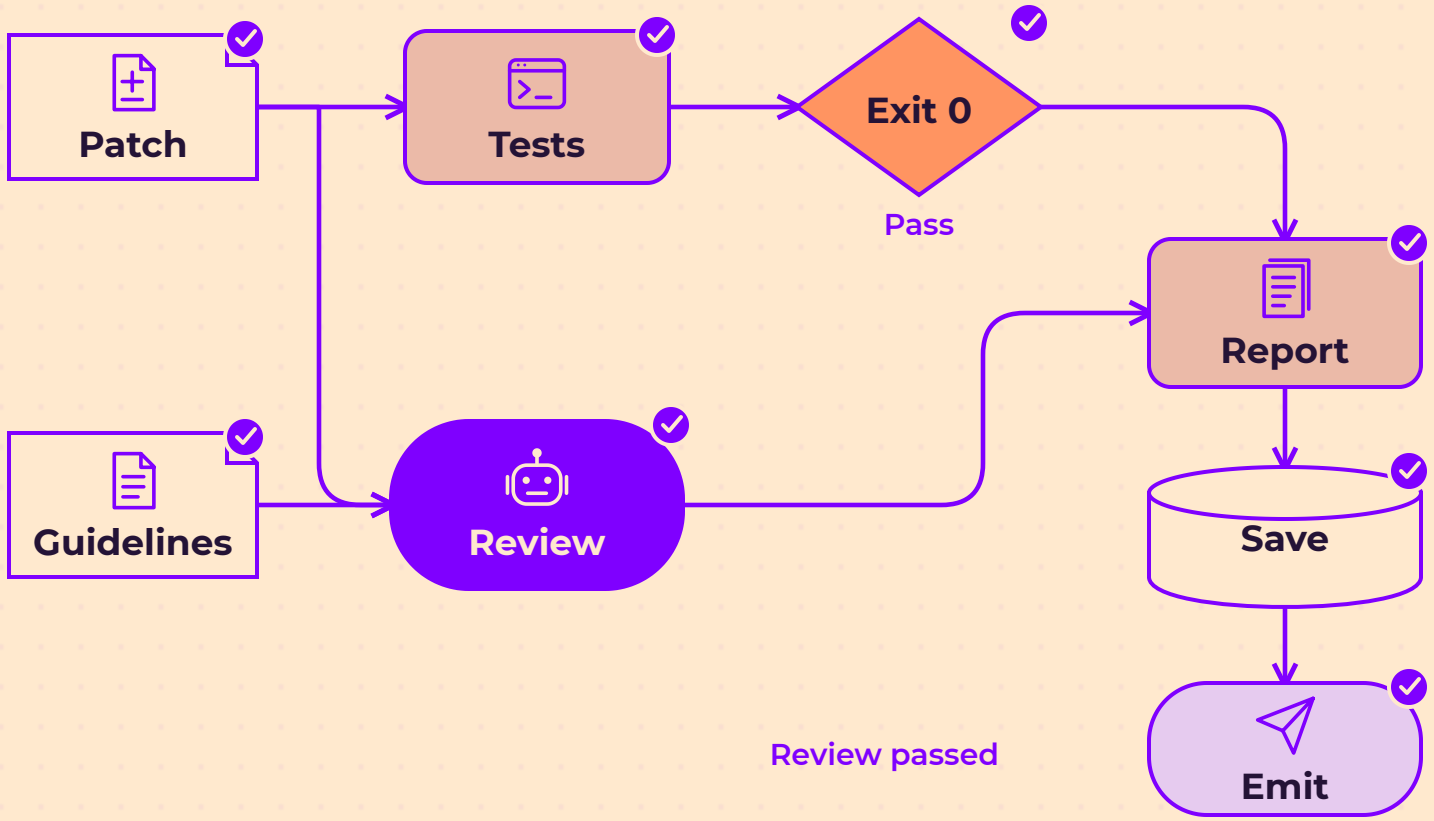


ILLUSTRATIVE DSL

```
flow("review", async () => {
  patch = input("patch")
  rules = input("guidelines@3")
  prompt = `Review:\n${rules}\n${patch}`
  tests = startTests(patch)
  review = agent("Ada", prompt)
  exit = await checkExit(tests)
  r = await join(exit, review)
  ok = r.complete && r.pass && r.review.ok
  report = makeReport(r, ok)
  saved = await kb.save(report)
  emit(saved.ref)
})
```

Confidential
© 2026 Mindera

Let’s see it run.



ILLUSTRATIVE DSL

```
flow("review", async () => {
  patch = input("patch")
  rules = input("guidelines@3")
  prompt = `Review:\n${rules}\n${patch}`
  tests = startTests(patch)
  review = agent("Ada", prompt)
  exit = await checkExit(tests)
  r = await join(exit, review)
  ok = r.complete && r.pass && r.review.ok
  report = makeReport(r, ok)
  saved = await kb.save(report)
  emit(saved.ref)
})
```

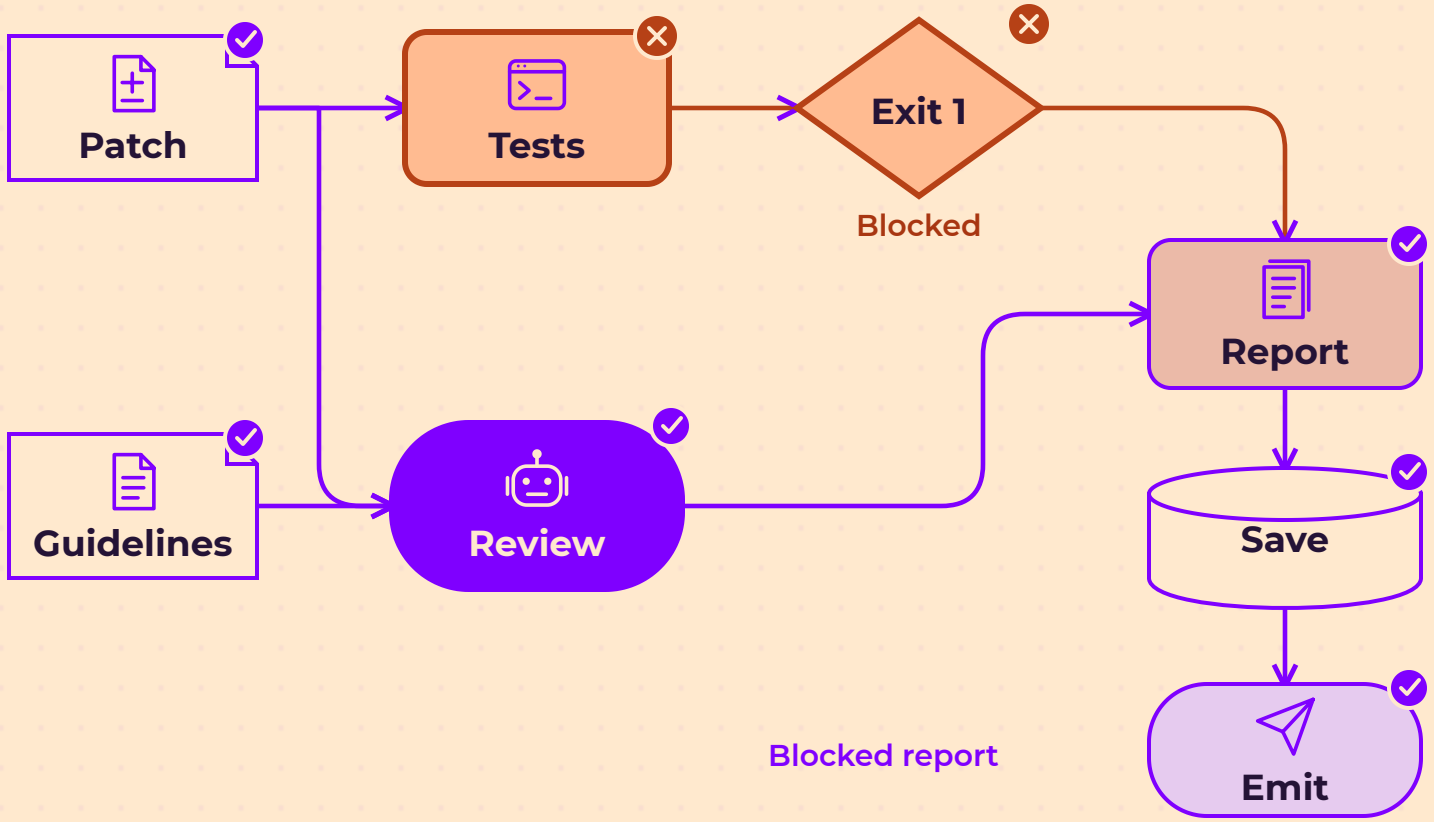
- Illustrative run

Reference emitted · review passed

Confidential

© 2026 Mindera

Now let’s make a test fail.



ILLUSTRATIVE DSL

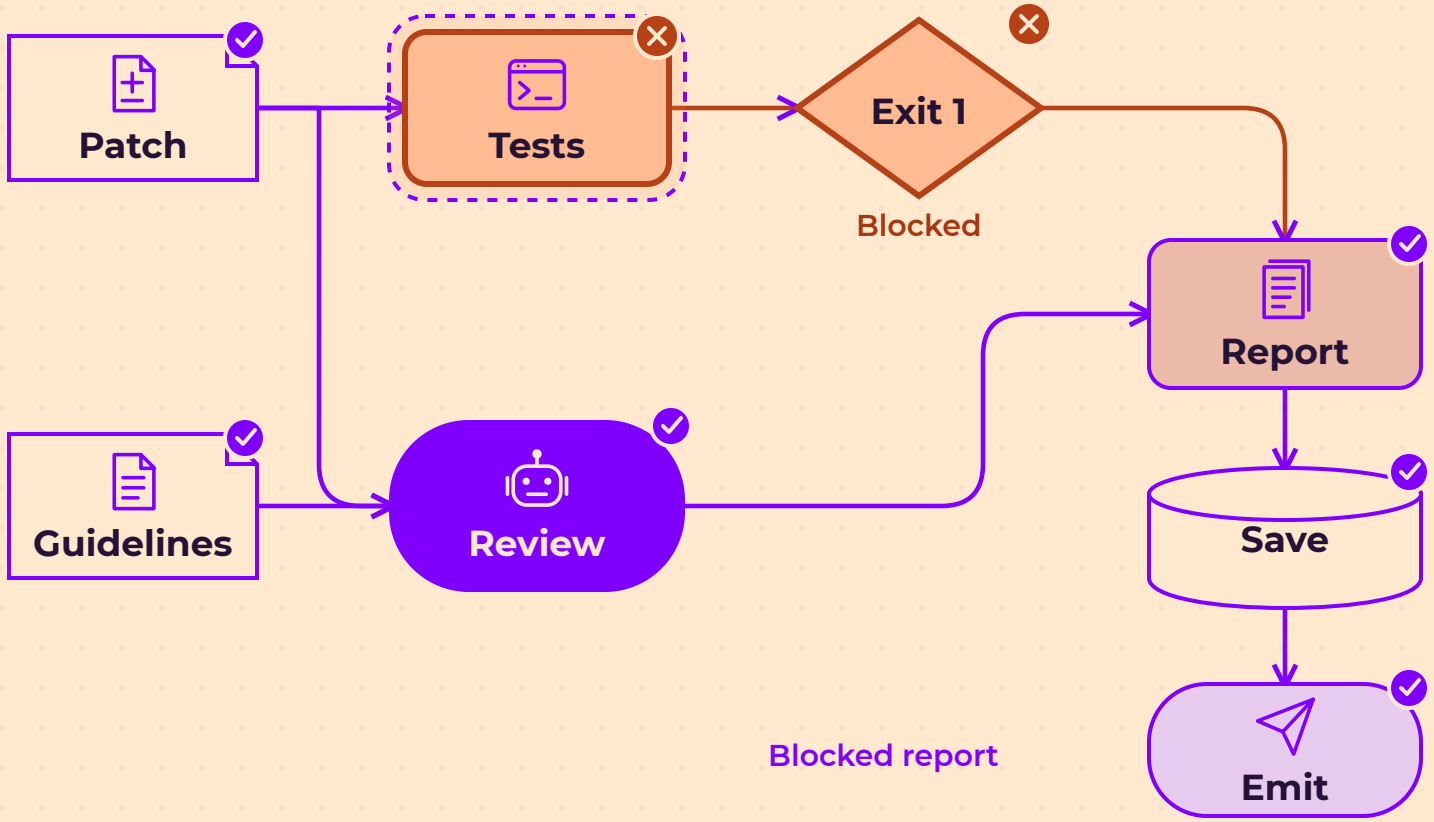
```
flow("review", async () => {
  patch = input("patch")
  rules = input("guidelines@3")
  prompt = `Review:\n${rules}\n${patch}`
  tests = startTests(patch)
  review = agent("Ada", prompt)
  exit = await checkExit(tests)
  r = await join(exit, review)
  ok = r.complete && r.pass && r.review.ok
  report = makeReport(r, ok)
  saved = await kb.save(report)
  emit(saved.ref)
})
```

- Illustrative run

Reference emitted · approval blocked

© 2026 Mindera

In order to understand, we need to be able to look inside.



ILLUSTRATIVE TRACE

Test runner

INPUT

```
commit a7f32d
```

ACTION

```
npm run test:required
```

FAILED SCENARIO

```
rejects cross-account order reads
```

ASSERTION

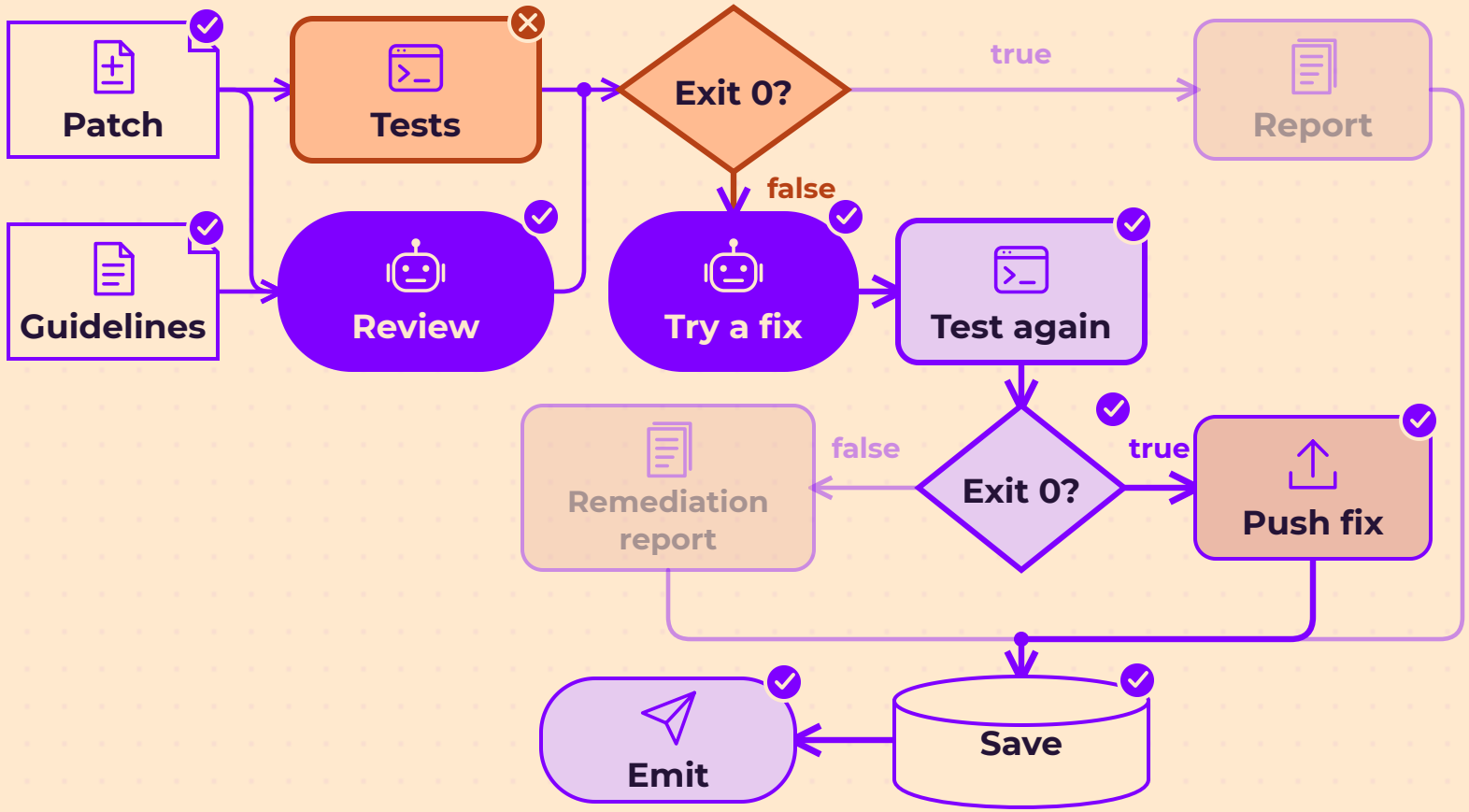
```
Expected 403; received 200
```

Select any node

Confidential

© 2026 Mindera

Then, we improve the workflow.



ILLUSTRATIVE DSL

EDIT → RUN

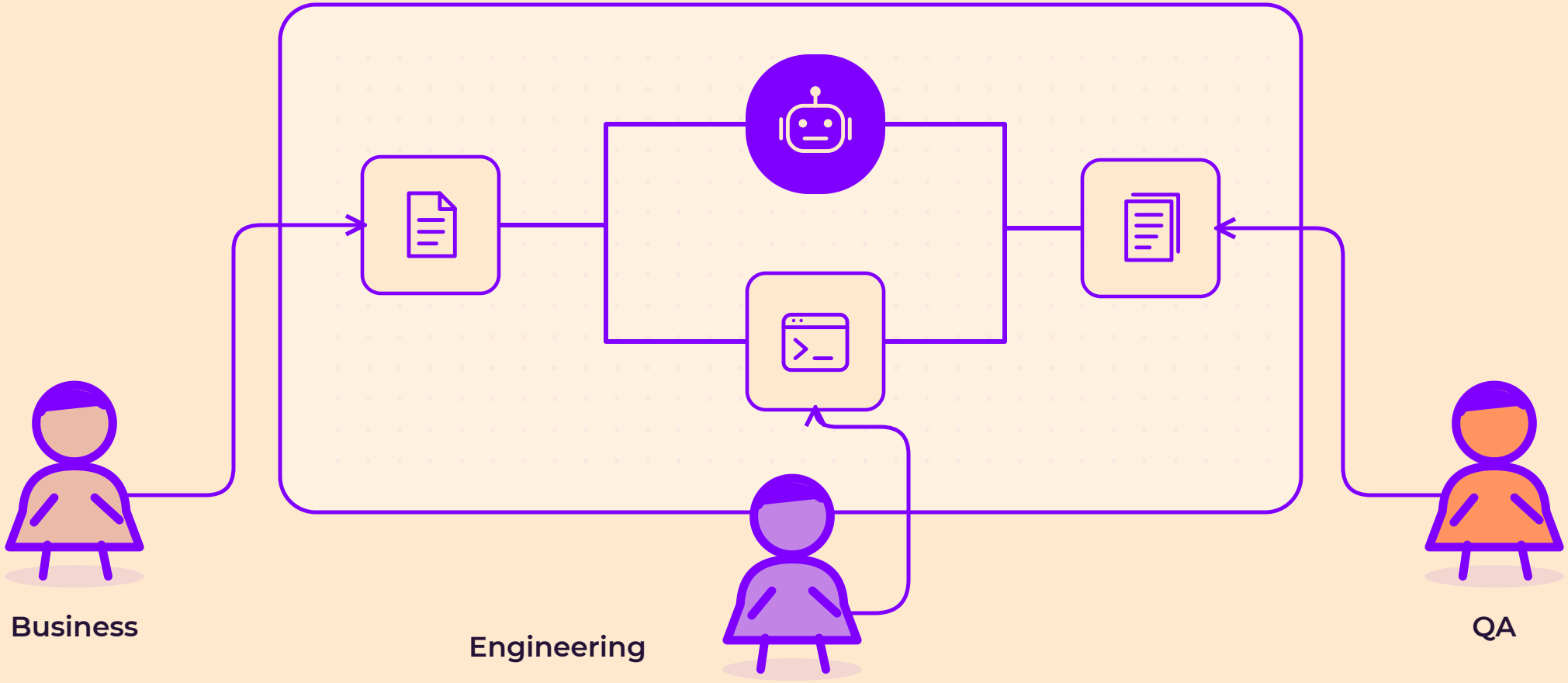
```
tests = await startTests(patch)
review = await agent("Ada", prompt)
if (tests.exit === 0) {
  result = makeReport(tests, review)
} else {
  fixPrompt = `Fix the failure:
    ${rules}\n${patch}\n${tests.output}`
  fix = await agent("Lin", fixPrompt)
  retest = await startTests(fix.patch)
  if (retest.exit === 0) {
    result = await push(fix.patch)
  } else {
    result = remediationReport(
      fix, retest
    )
  }
}
saved = await kb.save(result)
emit(saved.ref)
```

Fix pushed and saved.

Confidential

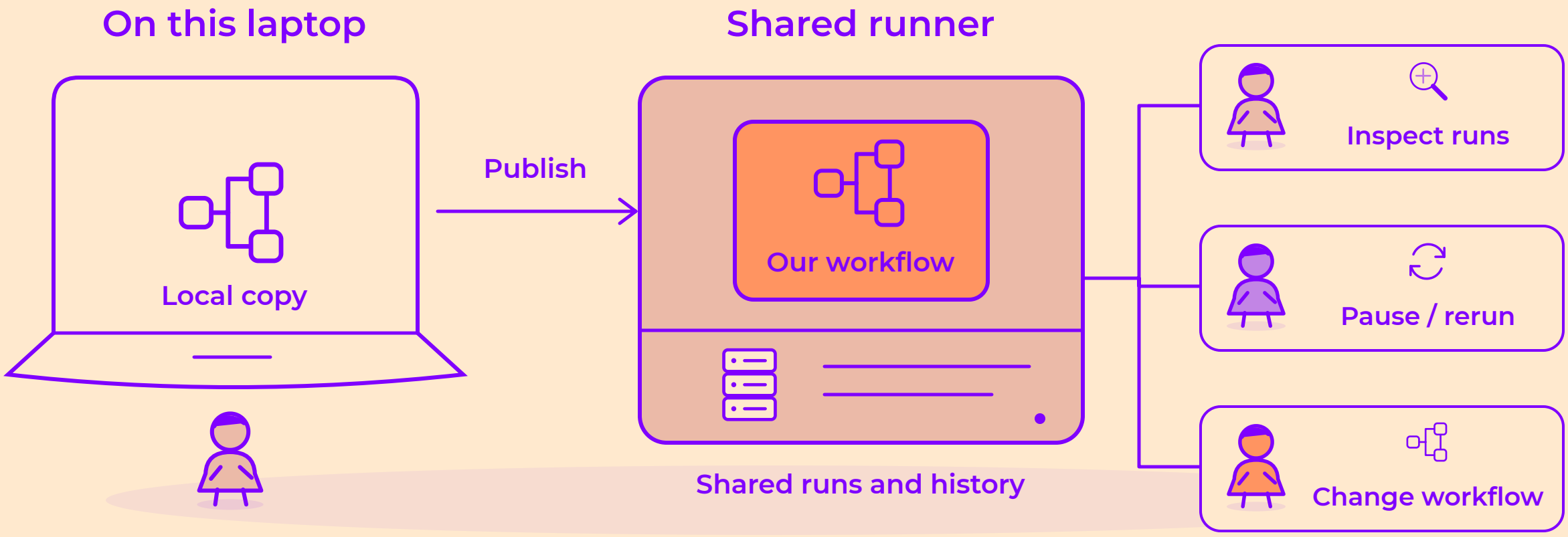
© 2026 Mindera

Hiding the complexity away means everyone can join in.



Confidential
© 2026 Mindera

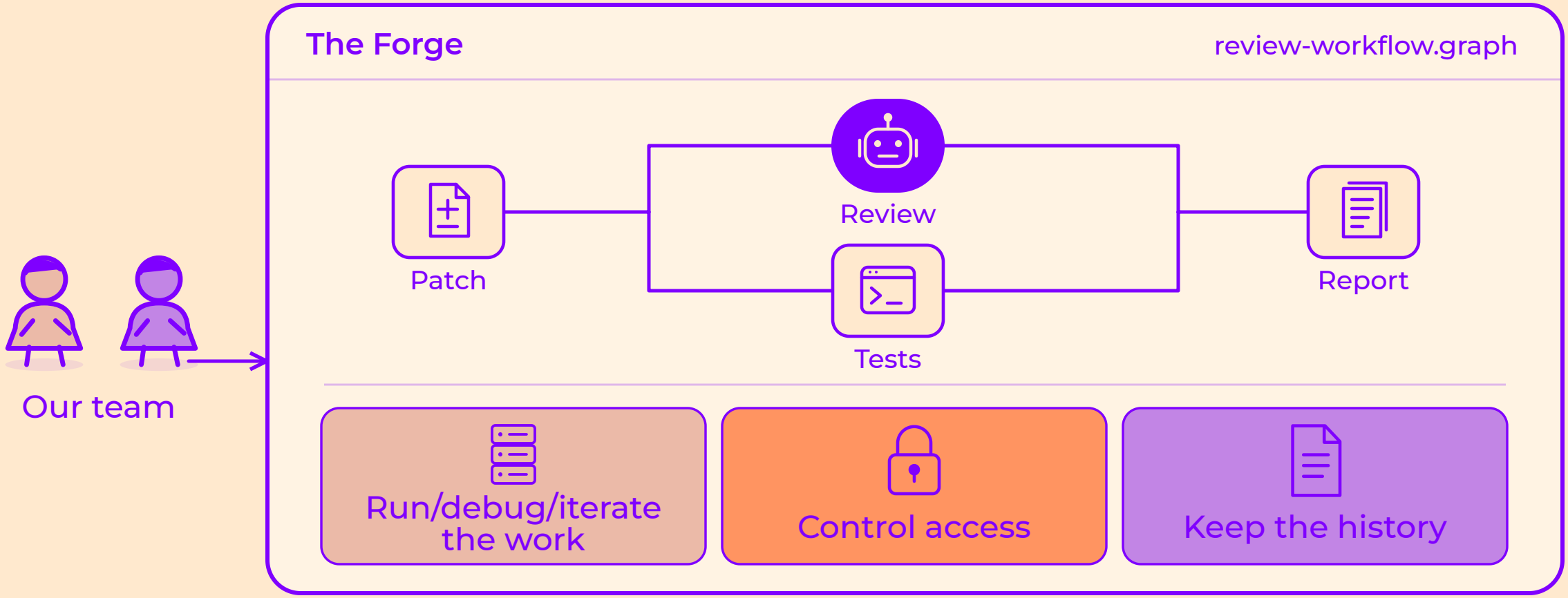
Where should we run it?



Everyone on the team can inspect and control it.

Confidential
© 2026 Mindera

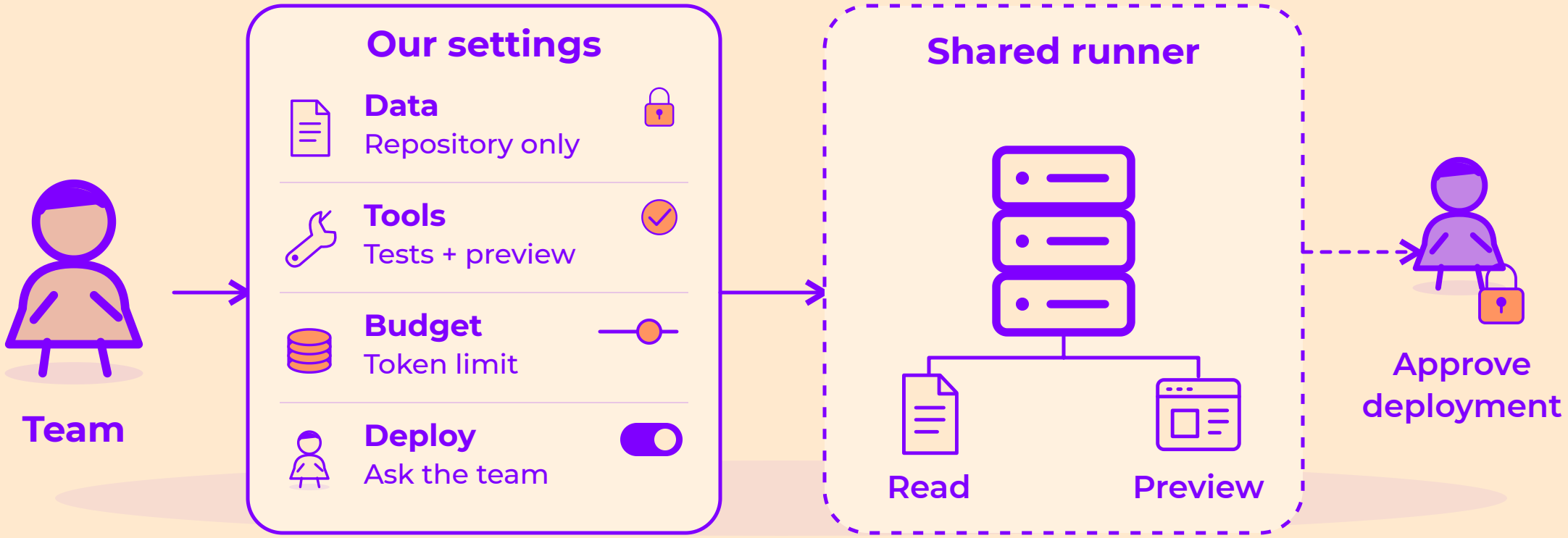
A forge for the team, owned by the team.



We can build on existing tools.

Confidential
© 2026 Mindera

Since we control it, we can keep settings tunable



The runtime enforces our choices.

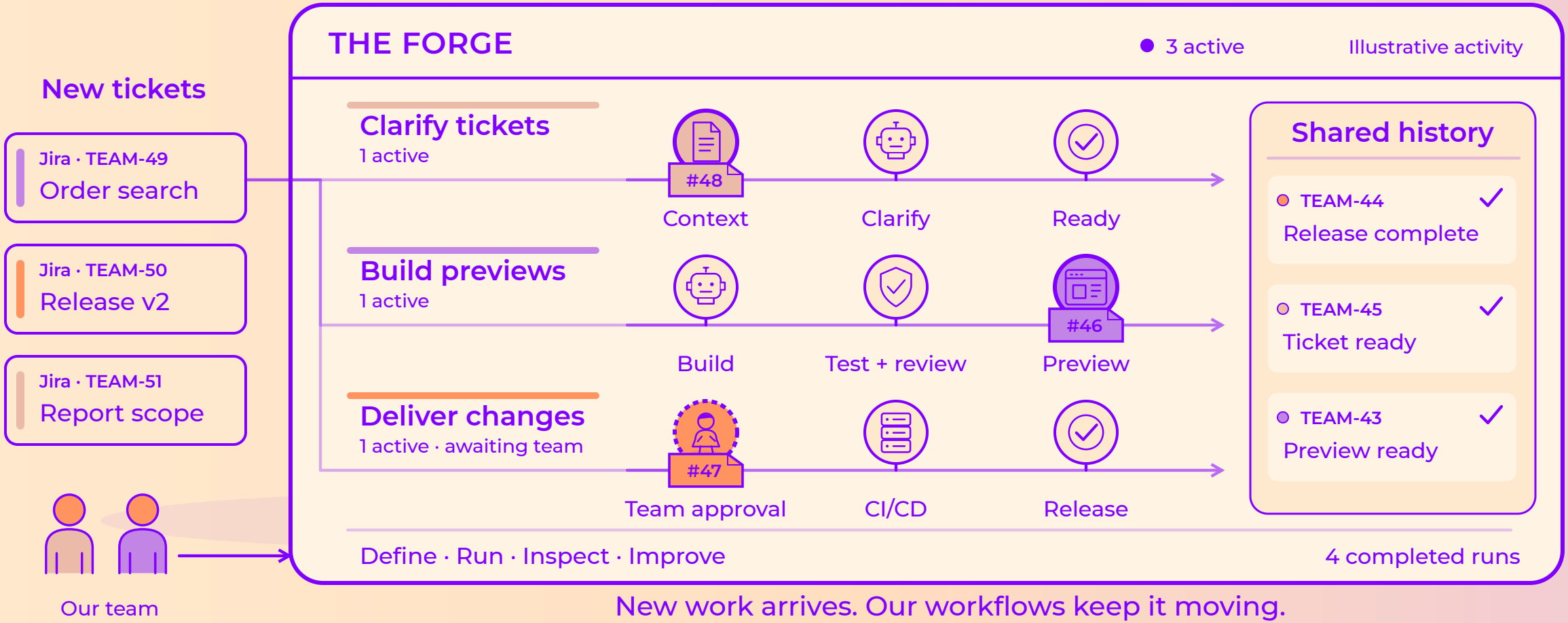
Confidential
© 2026 Mindera

It can also bring the team's knowledge together.



Confidential
© 2026 Mindera

Little by little, we step up the ladder.

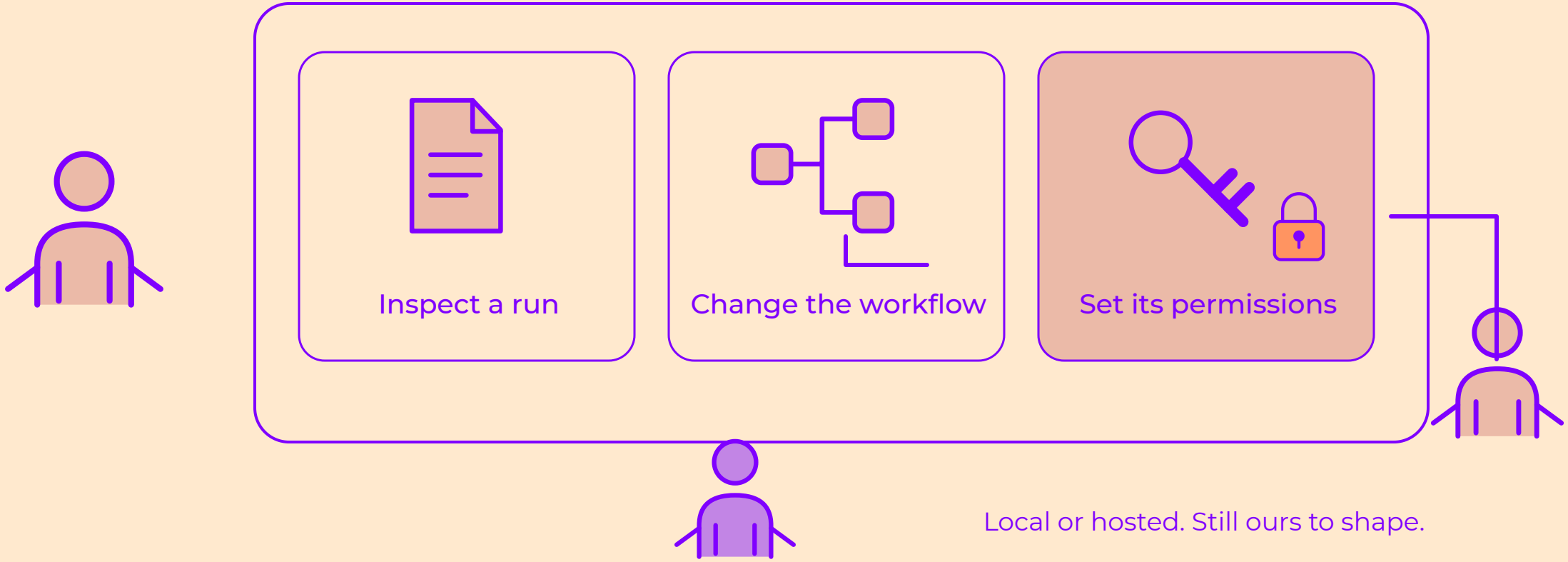


New work arrives. Our workflows keep it moving.

Confidential
© 2026 Mindera

No silver bullets, only extendability.

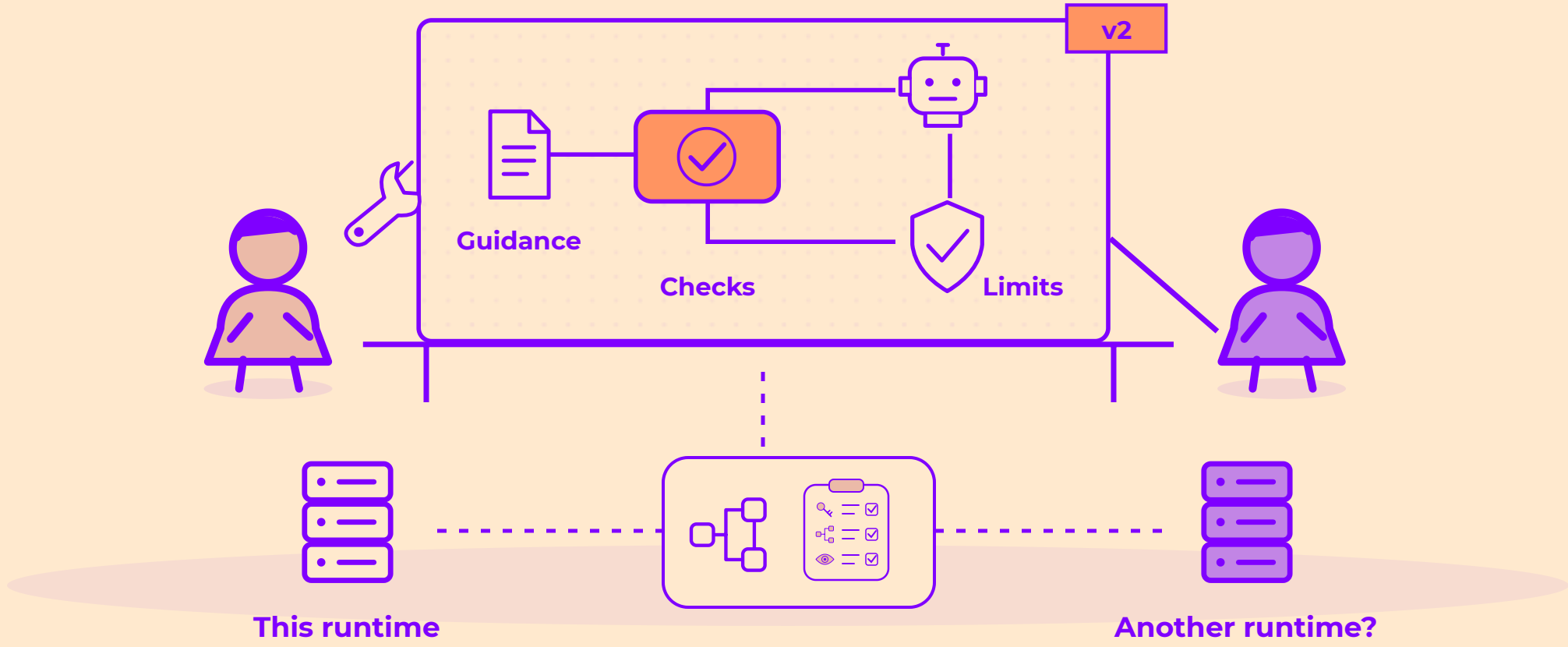
The people doing the work can shape the machinery.



Local or hosted. Still ours to shape.

Confidential
© 2026 Mindera

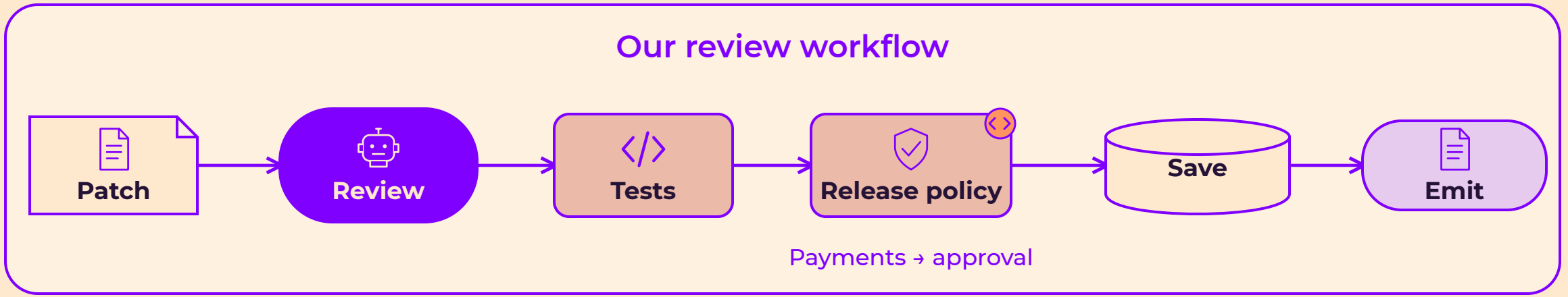
We can make changes and track them easily, and ship them too.



Confidential

© 2026 Mindera

As our work changes, we can add to the machinery...

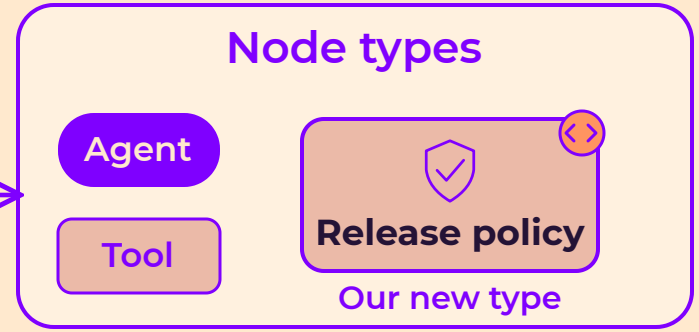


Write custom behaviour

Our team

```
release-policy.ts
defineNode('release-policy', {
  run: patch => ({
    needsApproval: touches(patch, 'payments/')
  })
})
```

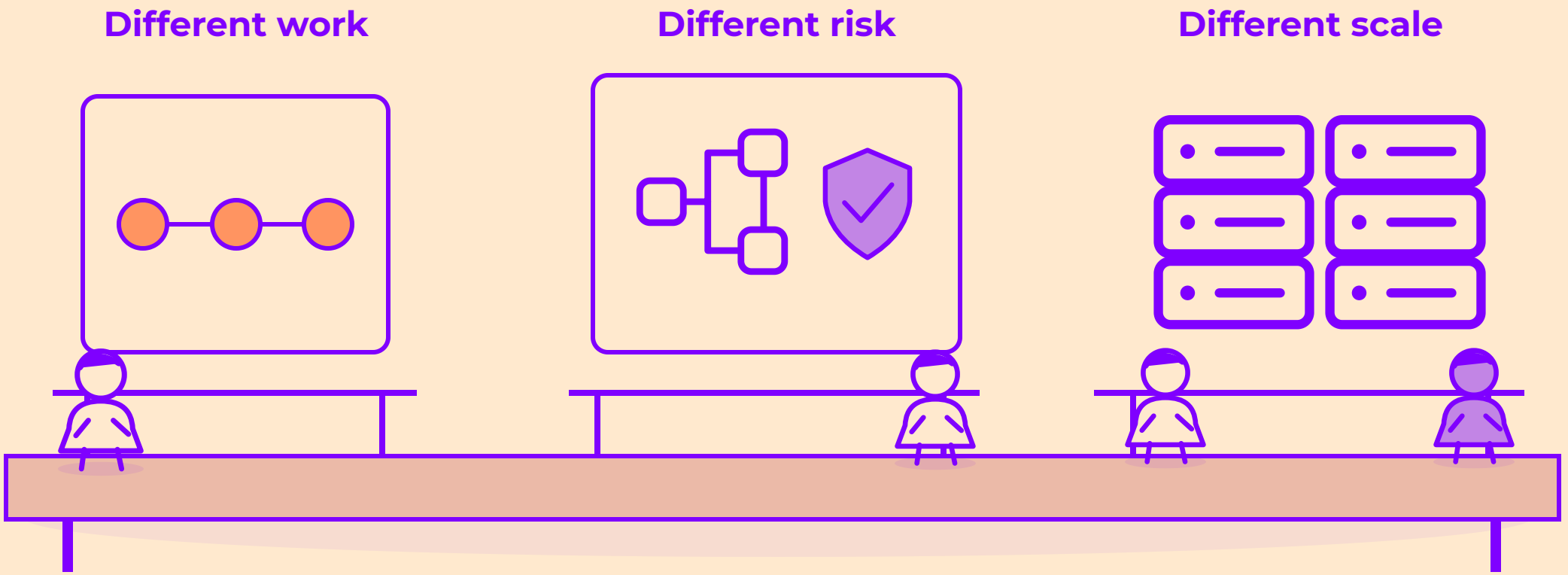
Register



Illustrative node API

Confidential
© 2026 Mindera

...as every team will need something different.



Confidential

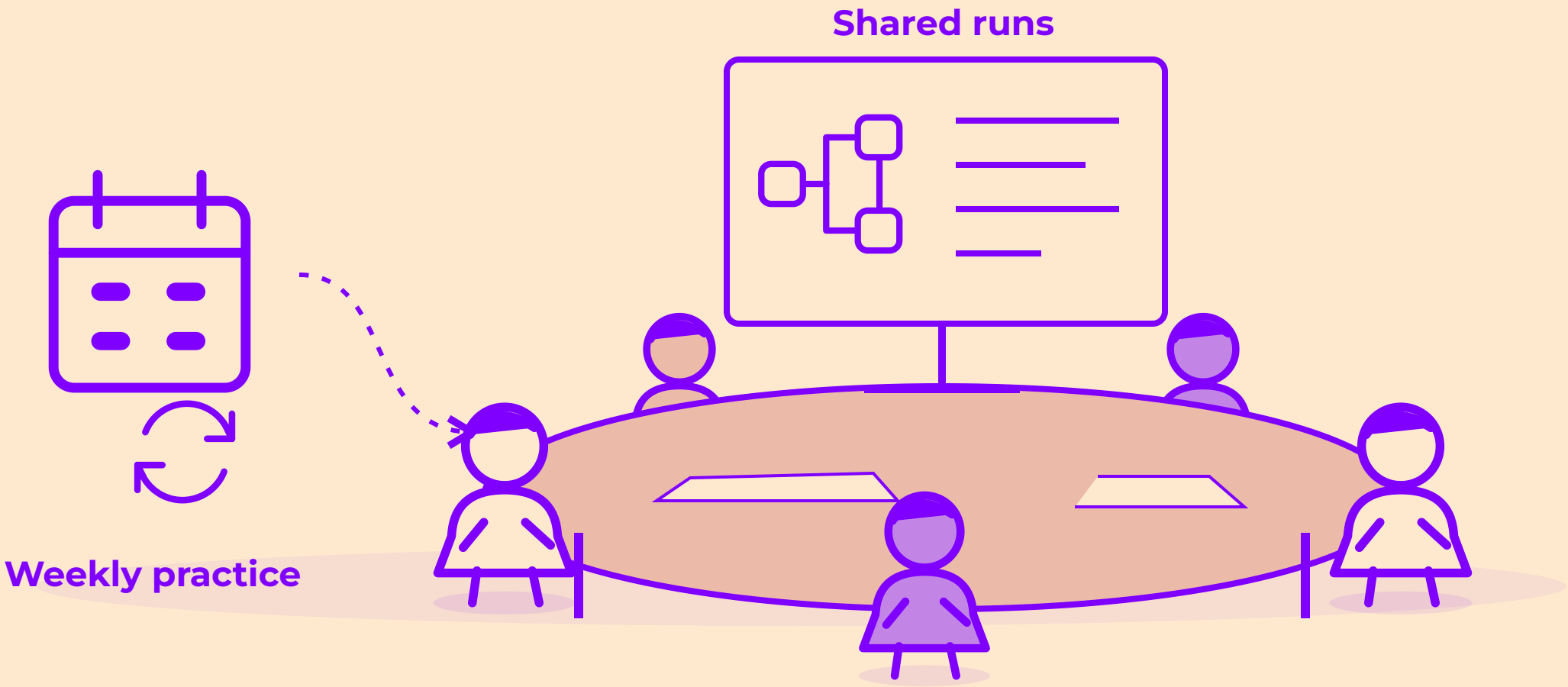
© 2026 Mindera

How do we approach this, as a team?

Confidential

© 2026 Mindera

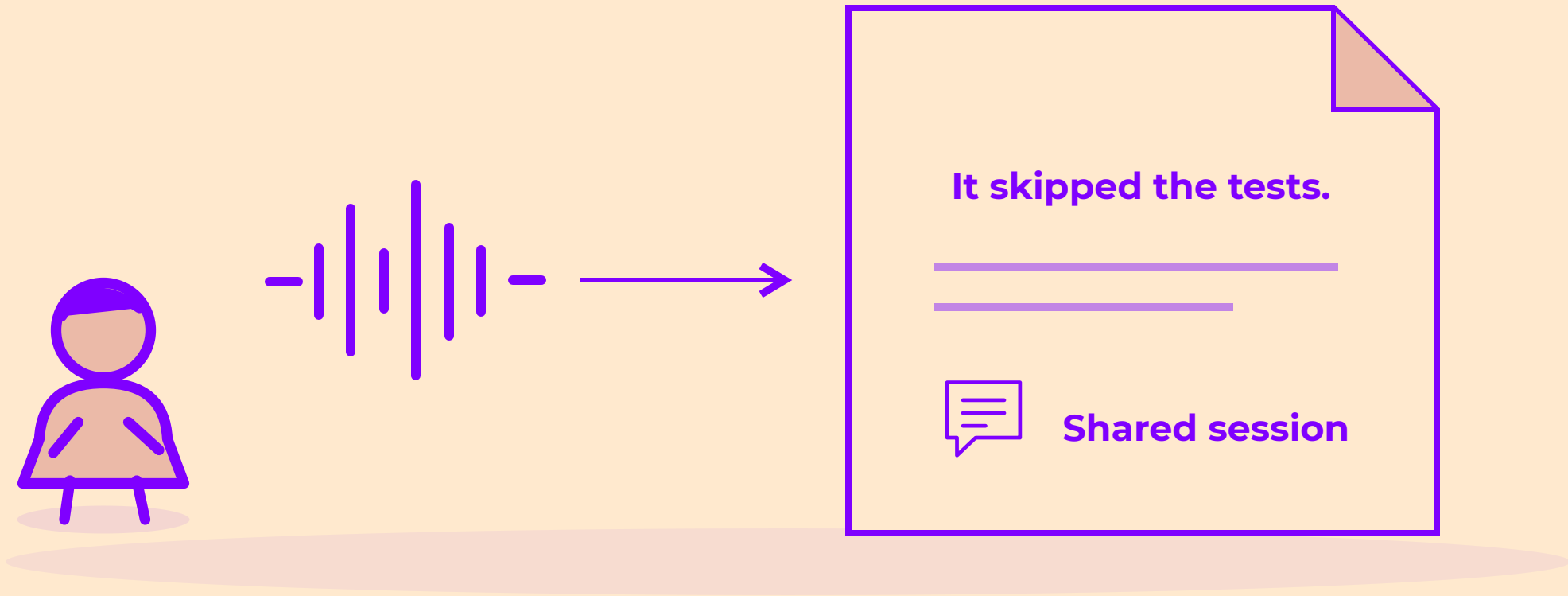
Make shared learning a team ritual.



Confidential

© 2026 Mindera

Capture the problem in their own words.

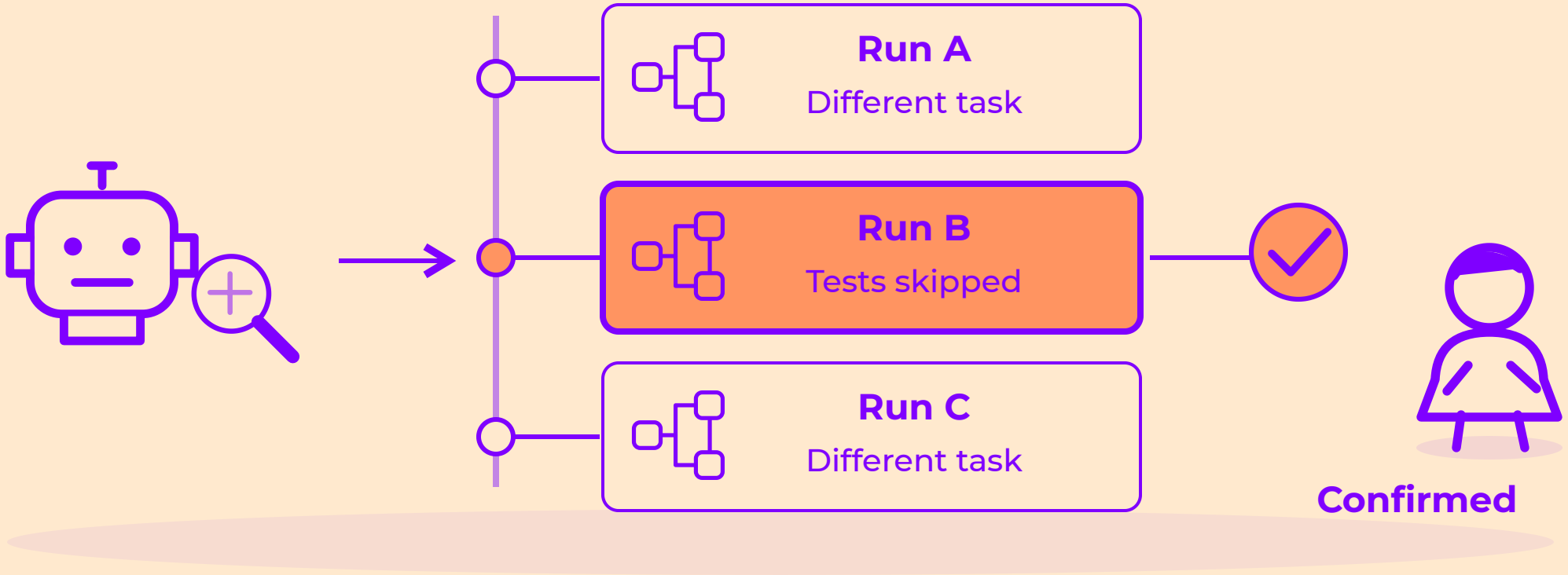


Illustrative team review

Confidential

© 2026 Mindera

Find the run where it happened.

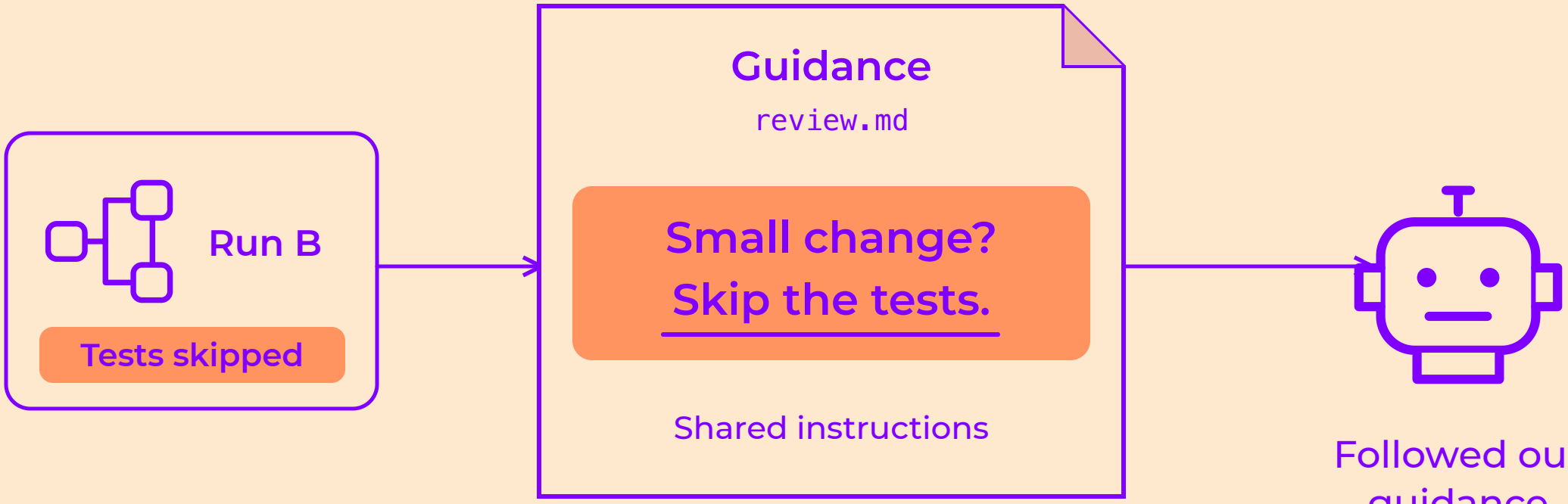


Illustrative team review

Confidential

© 2026 Mindera

Why did it behave that way?



The instruction was wrong.

Illustrative team review

Confidential
© 2026 Mindera

What should we do differently next time?

Guidance

`review.md`

Small change?
~~**Skip the tests.**~~

Shared instructions

The team agrees

Run tests for every change.



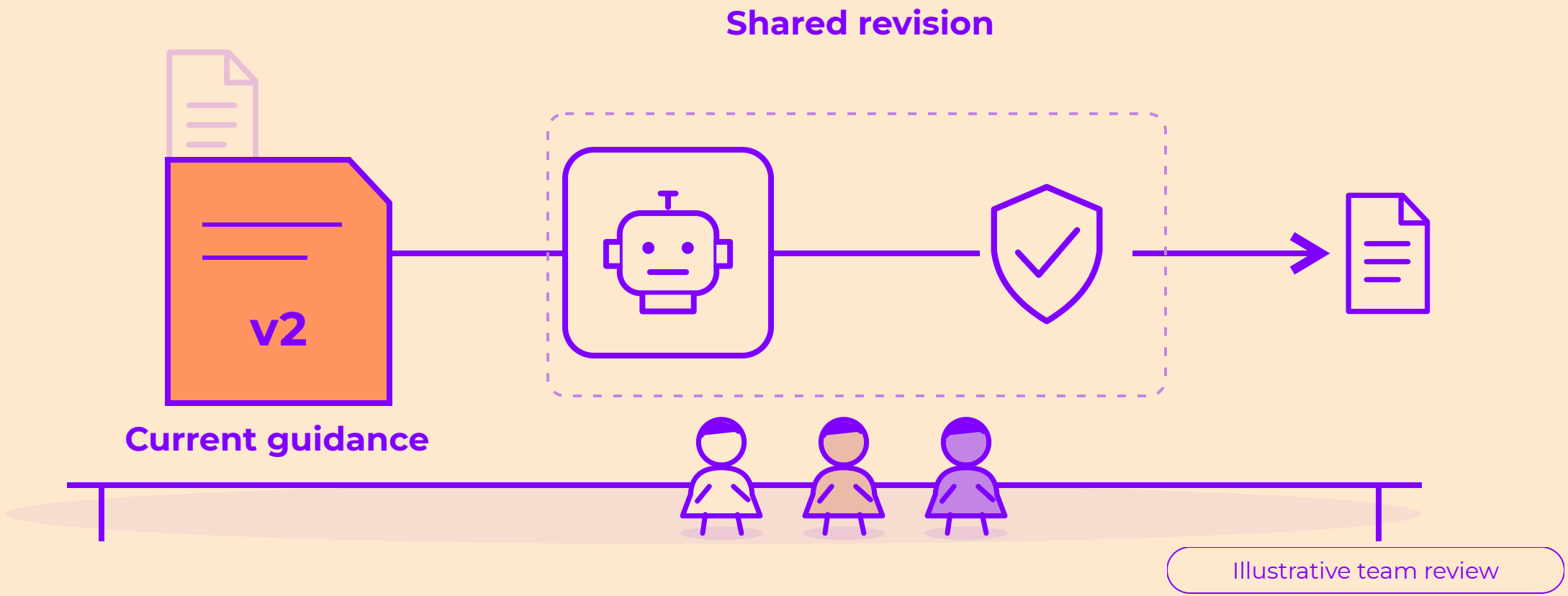
Change our shared guidance.

Illustrative team practice

Confidential

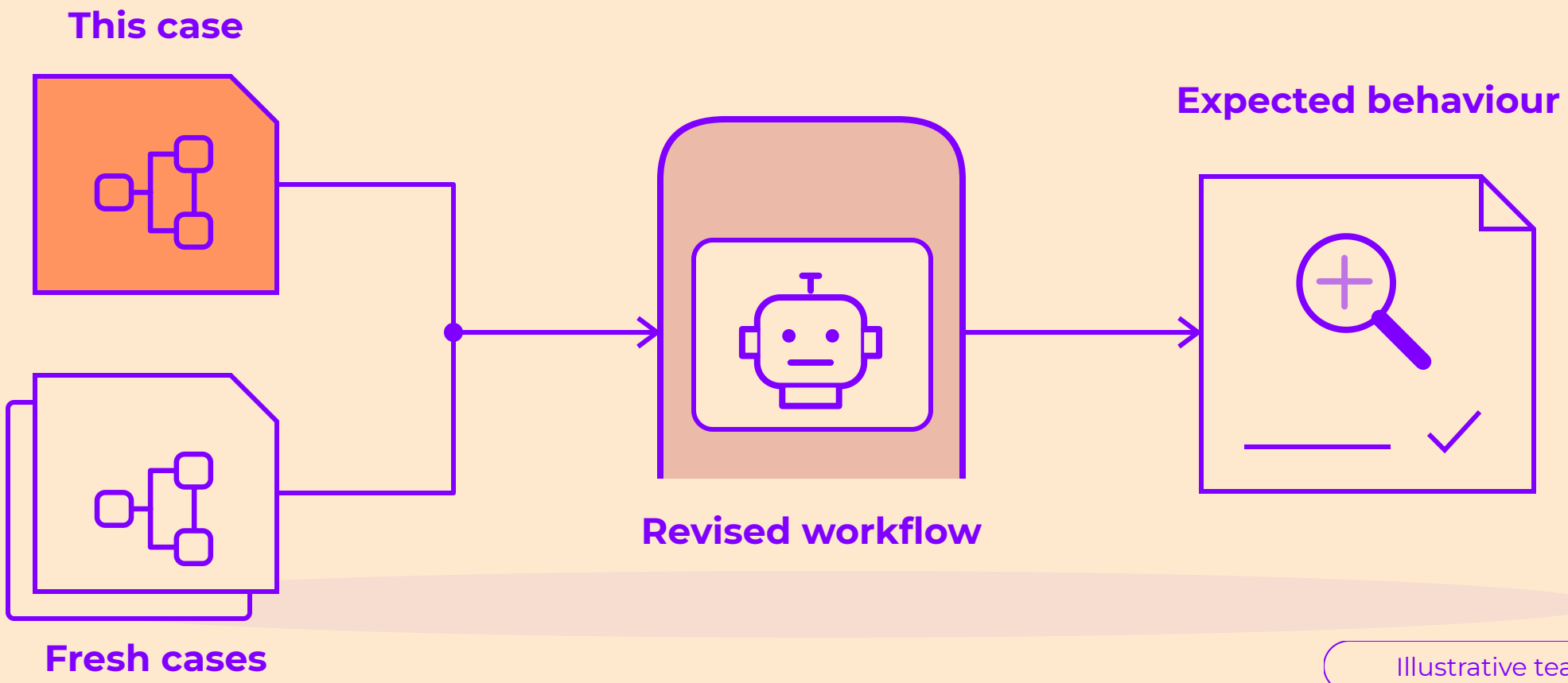
© 2026 Mindera

Fix the part that caused the problem.



Confidential
© 2026 Mindera

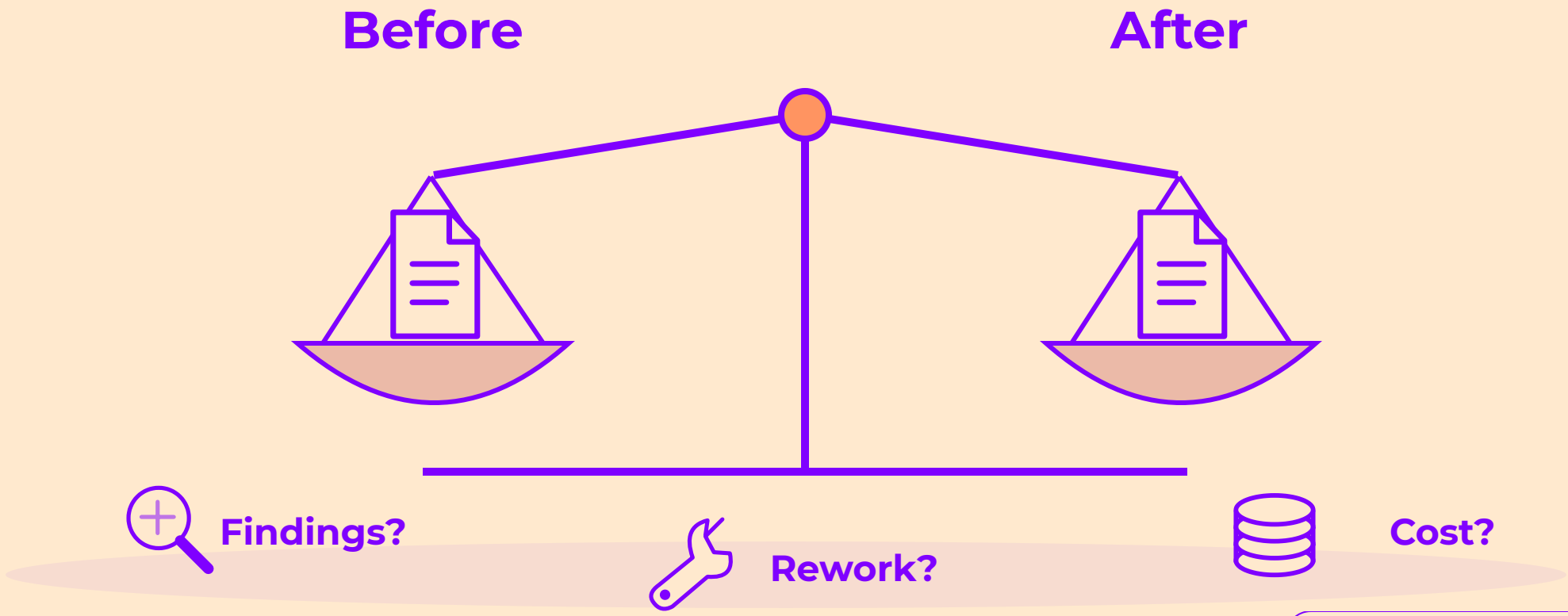
We measure, instrument, make changes and observe.



Illustrative team review

Confidential
© 2026 Mindera

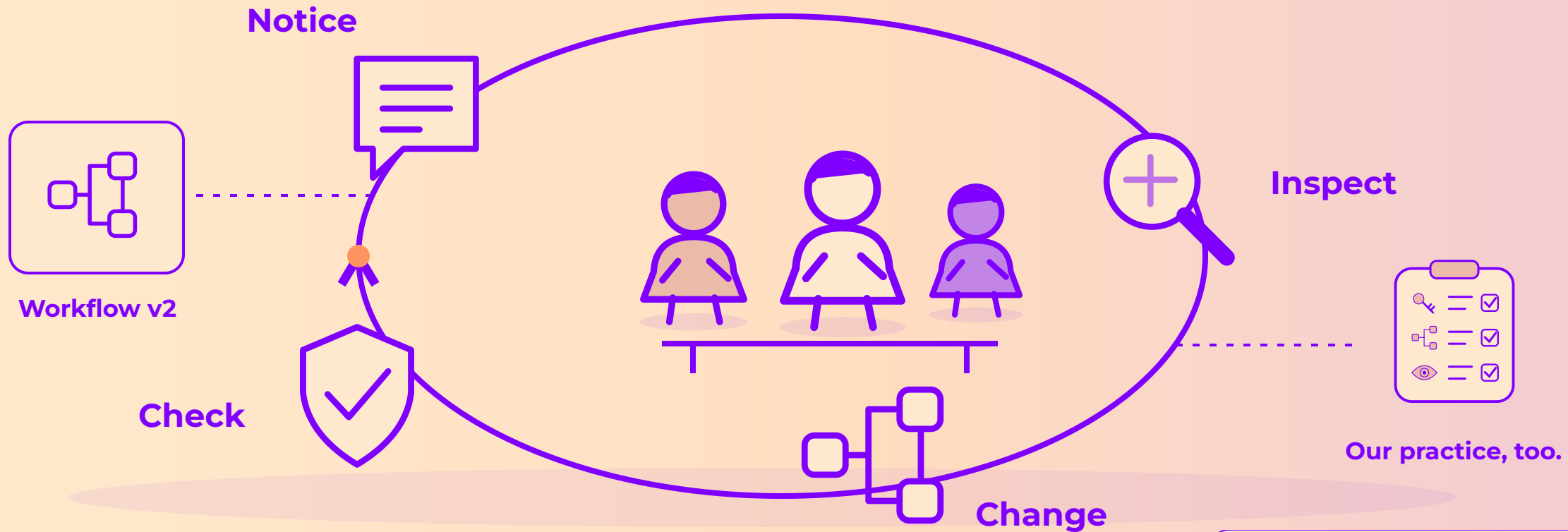
Did our change actually help?



Illustrative team review

Confidential
© 2026 Mindera

Then do it again, together.



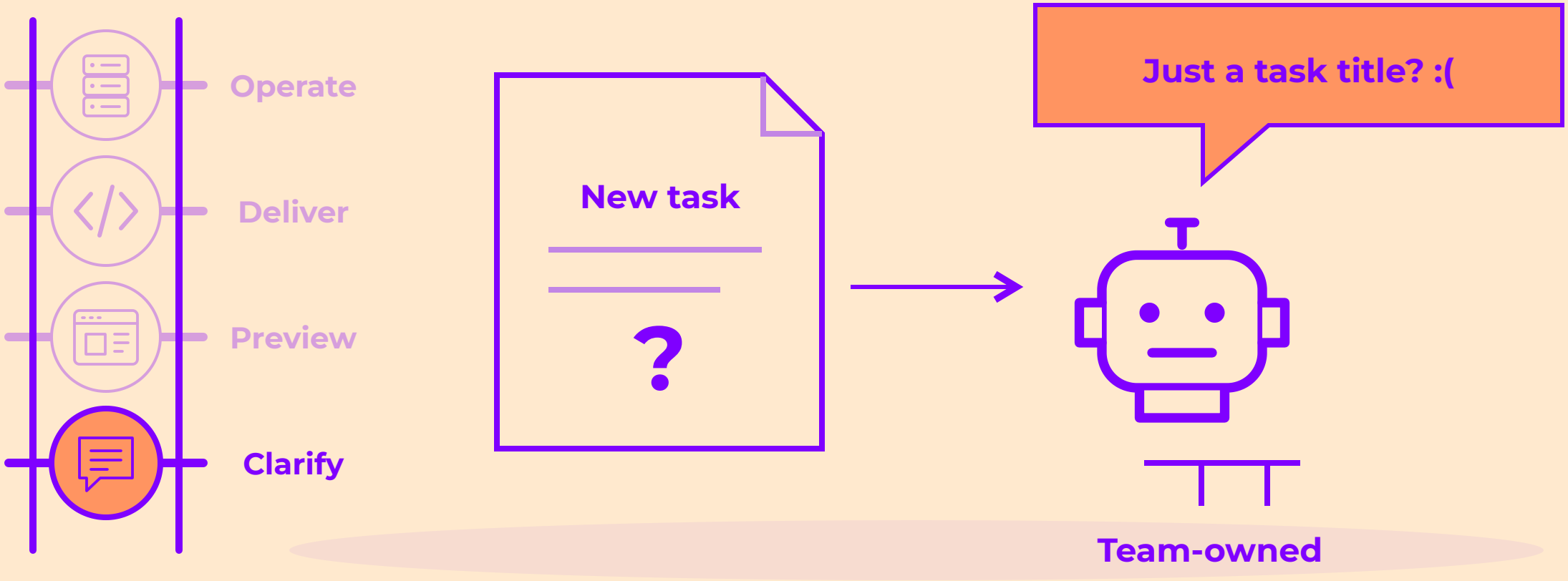
Illustrative team review

Confidential
© 2026 Mindera

What does this ladder look like in practice?

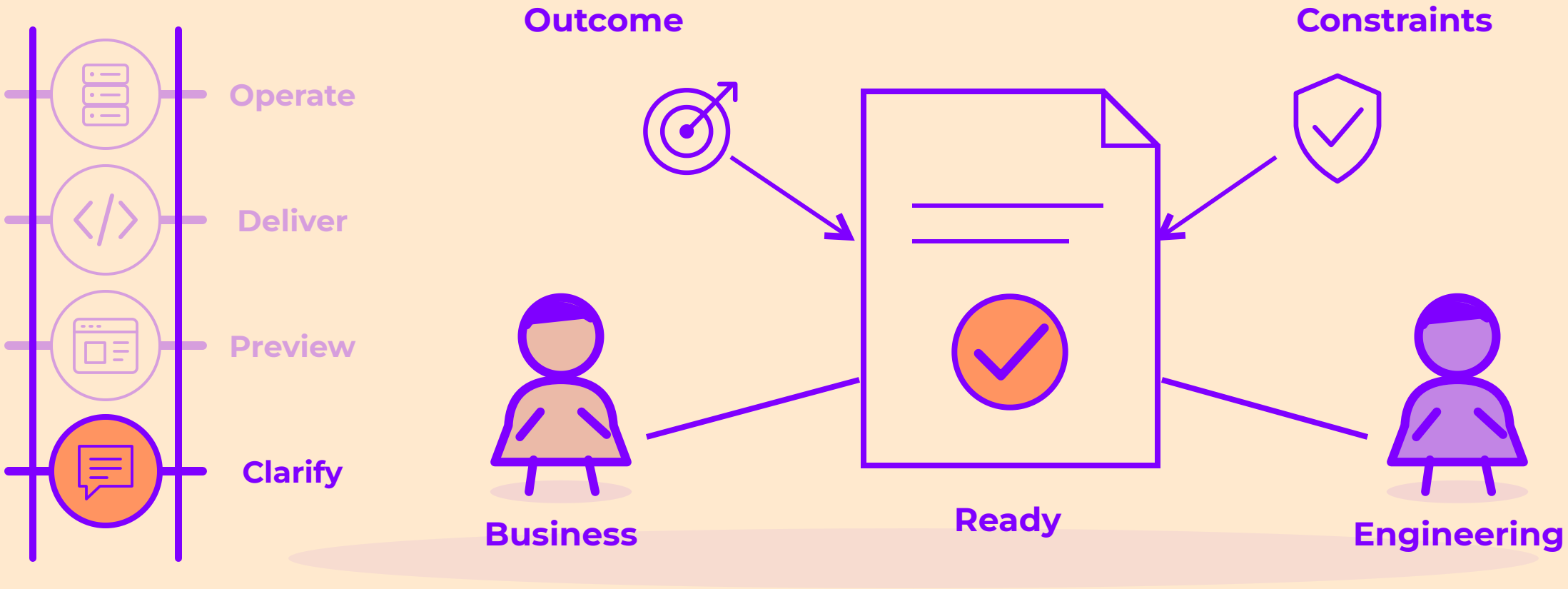
Confidential © 2026 Mindera

Start with something to ease task management.



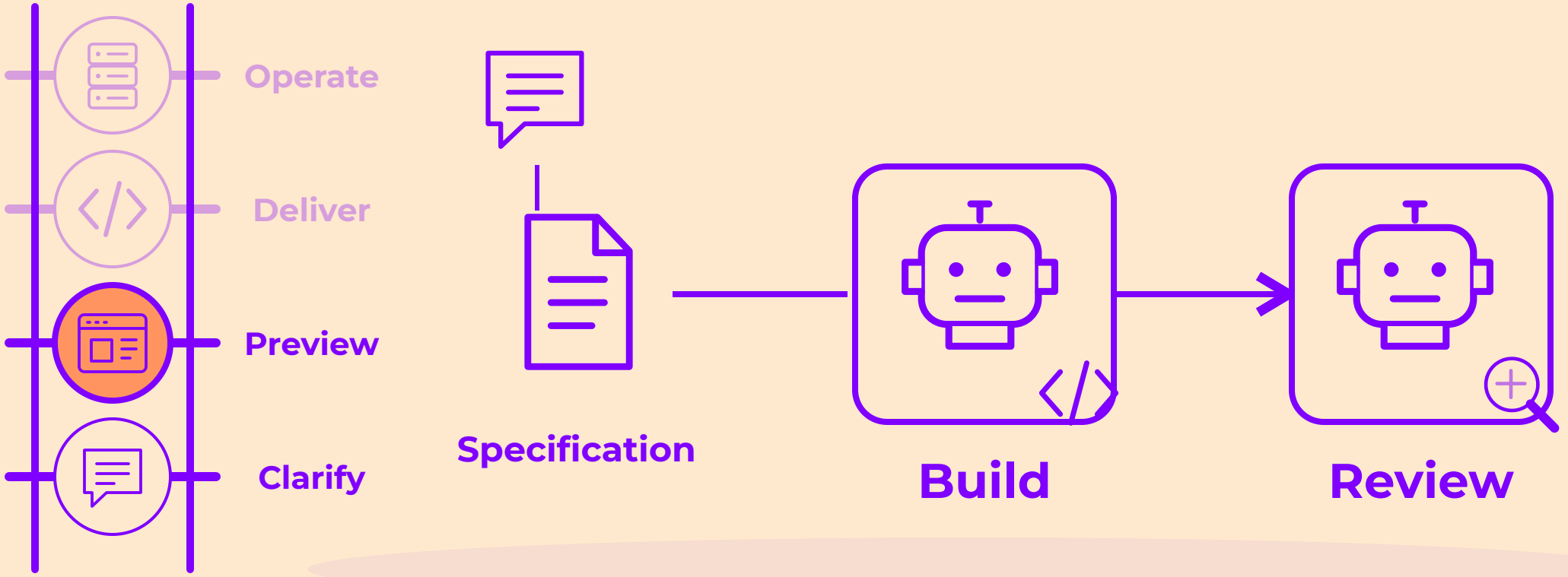
Confidential
© 2026 Mindera

Agree together what “ready” means, build, and dogfood.



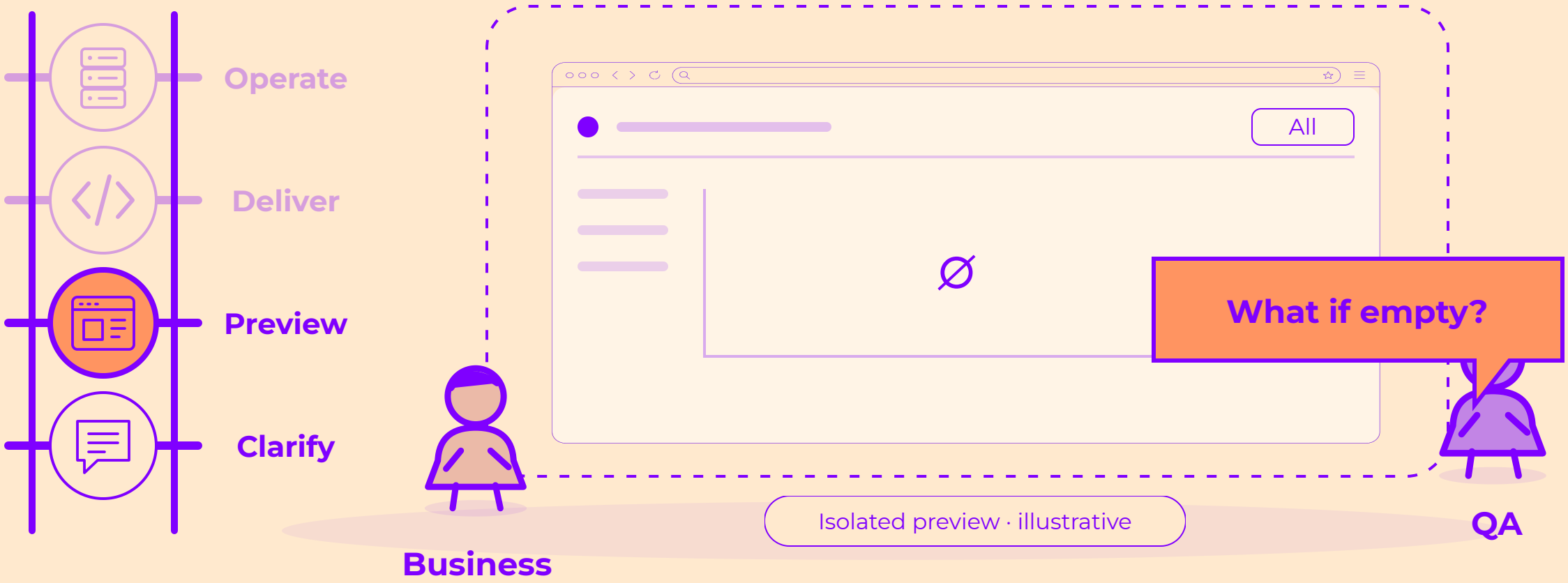
Confidential
© 2026 Mindera

Then experiment with first-draft feature agents.



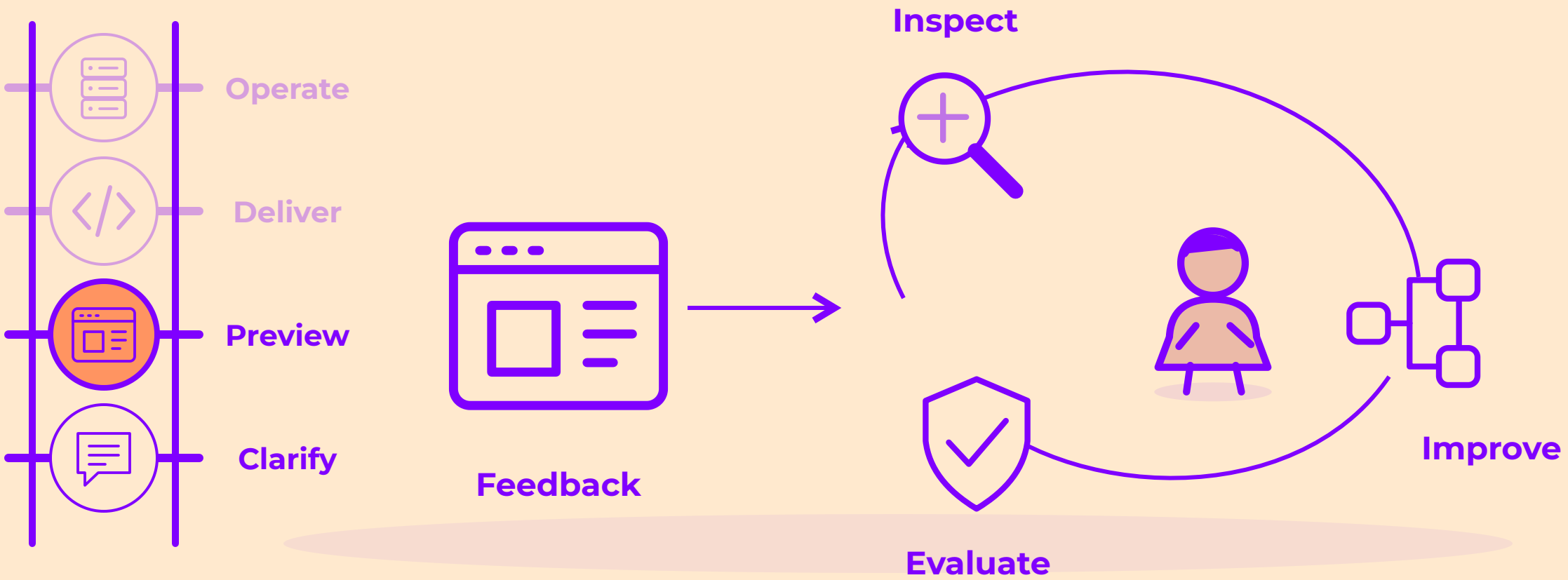
Confidential
© 2026 Mindera

Give people something they can try.



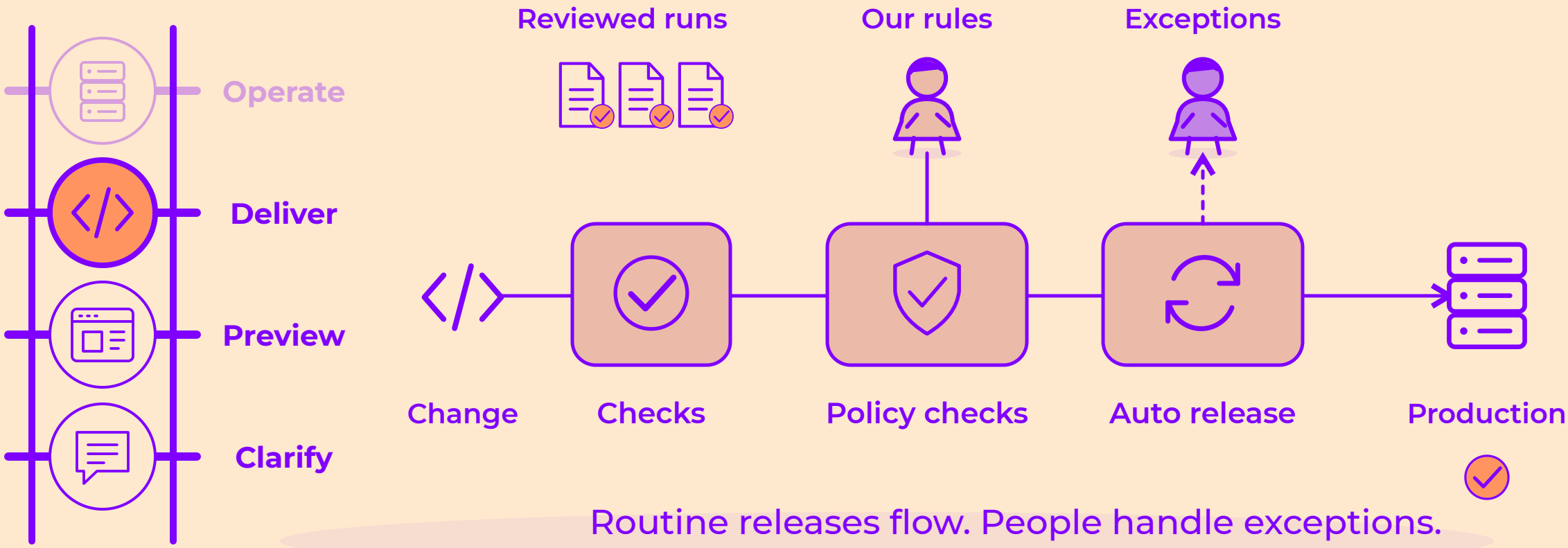
Confidential
© 2026 Mindera

Use the same loop to make it better.



Confidential
© 2026 Mindera

As trust grows, we can automate real releases.



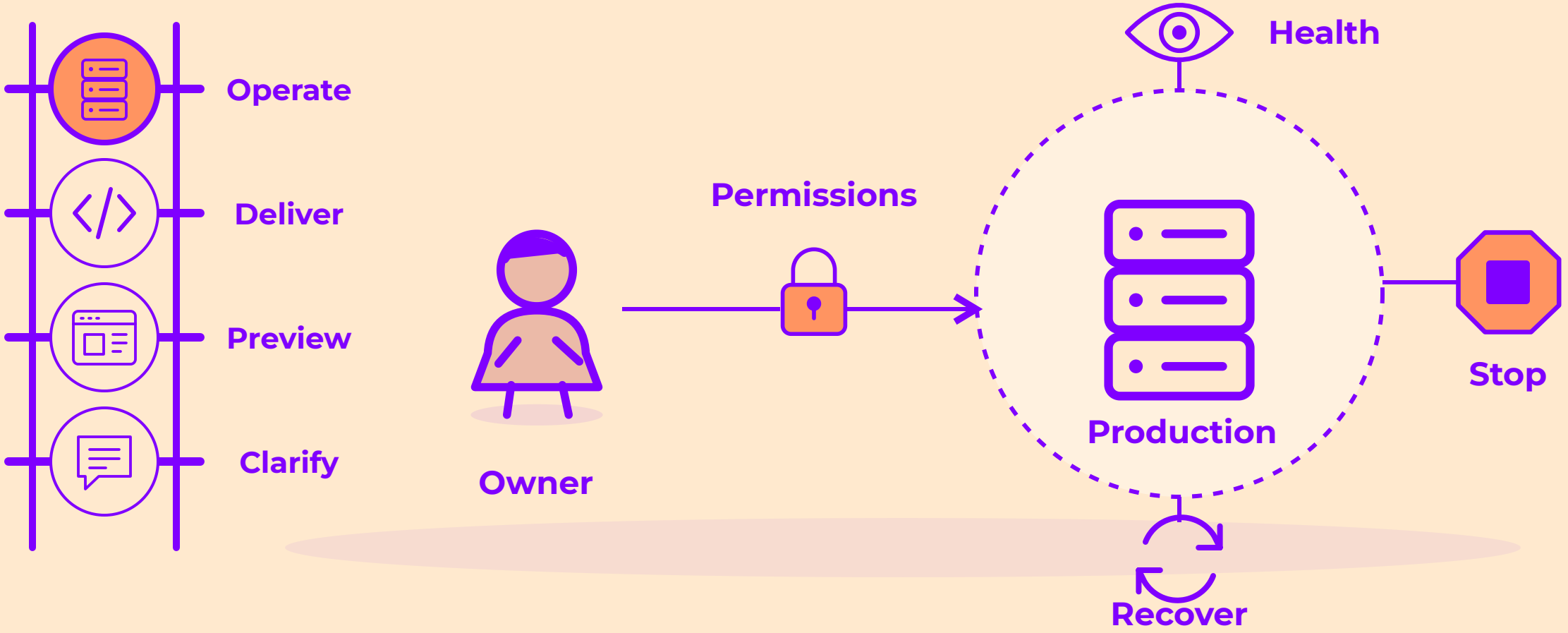
Routine releases flow. People handle exceptions.

For the release path we know and trust.

Confidential

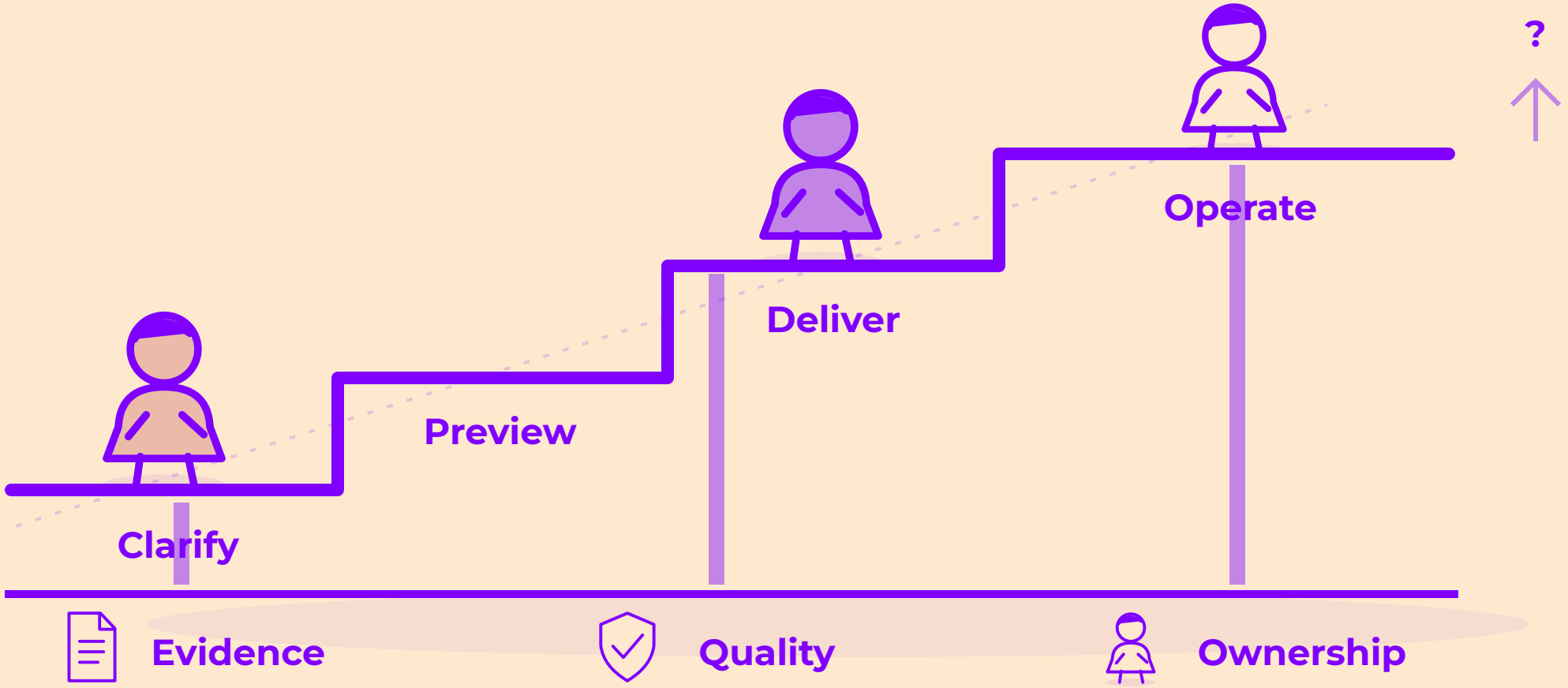
© 2026 Mindera

Give it a clear job in production.



Confidential
© 2026 Mindera

What makes us ready for the next step?

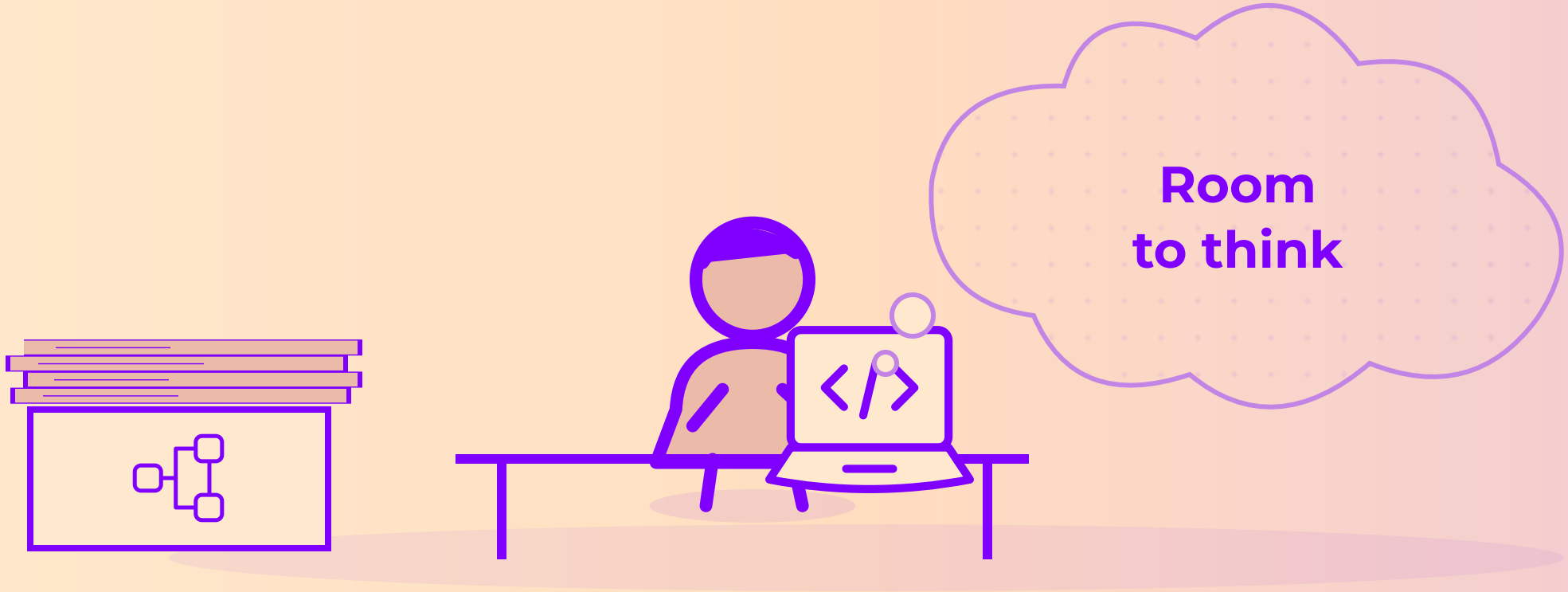


Confidential

© 2026 Mindera

What could we do with that time?

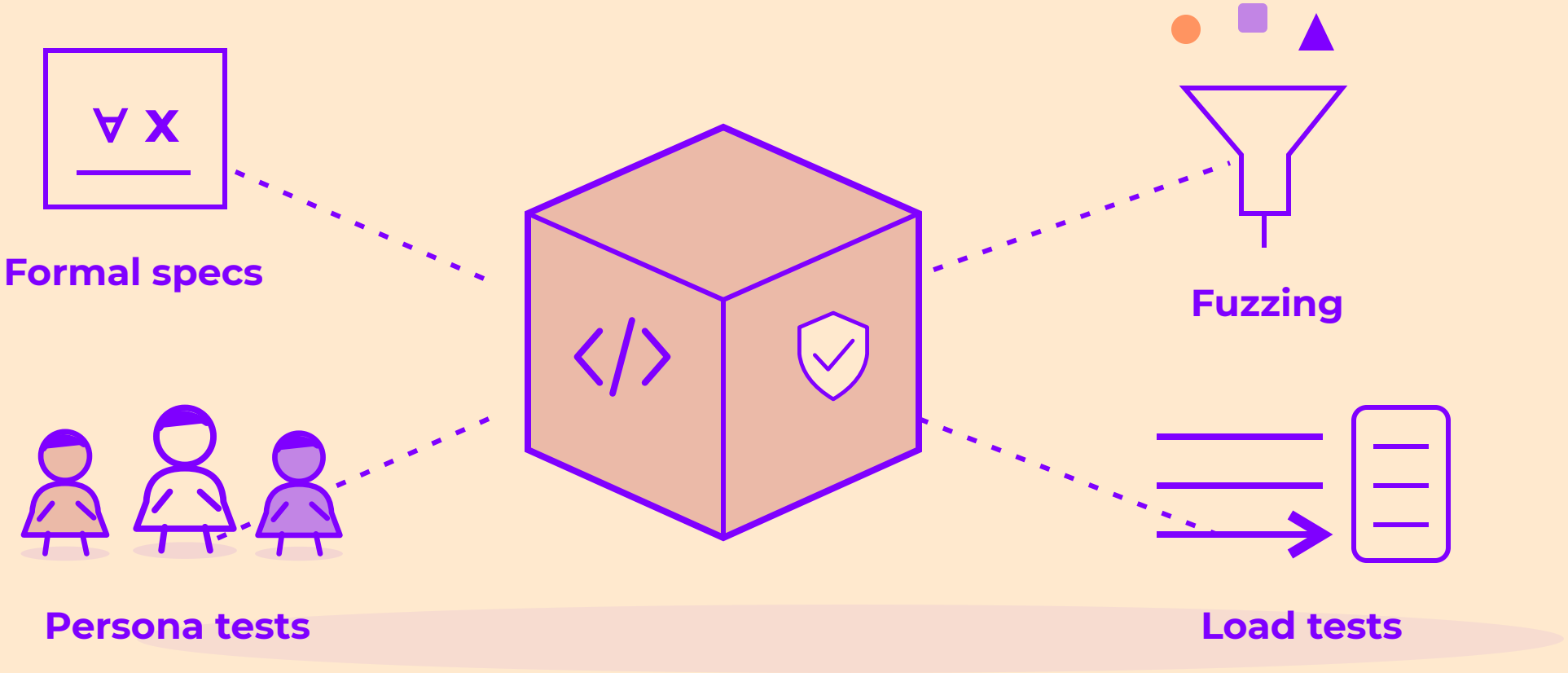
If routine effort falls...



Confidential

© 2026 Mindera

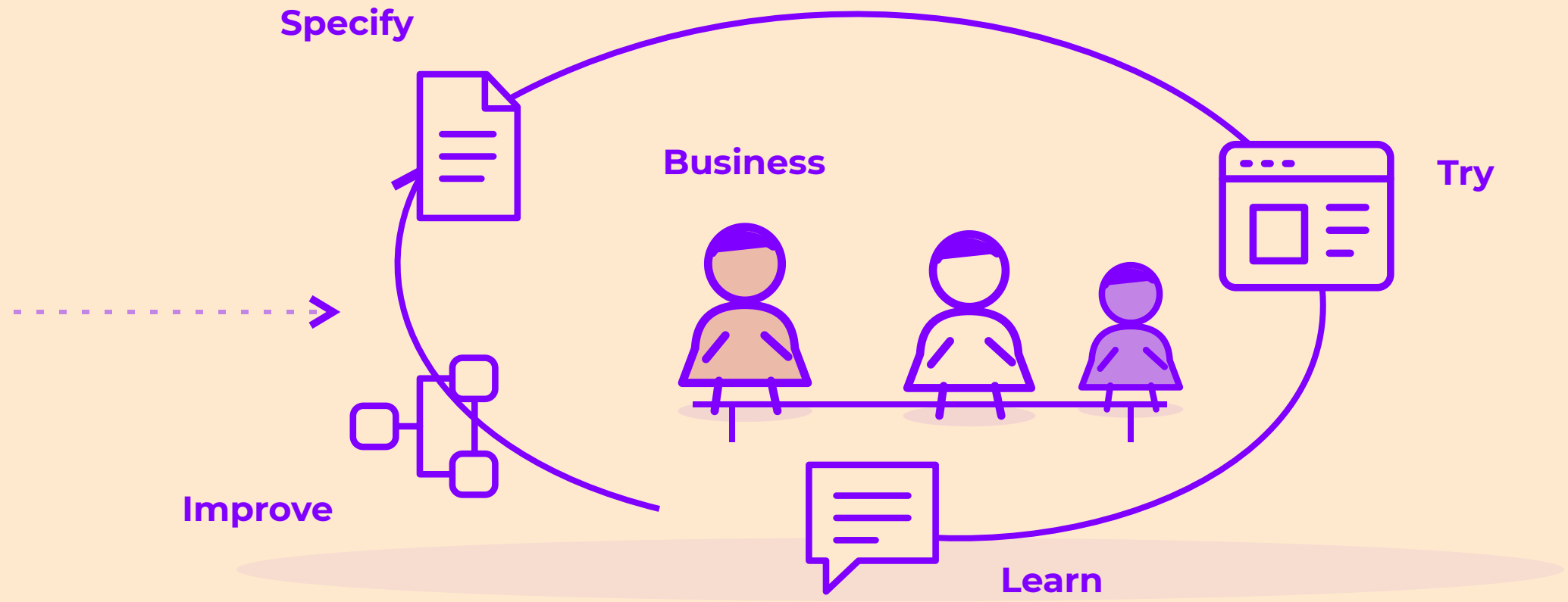
We could test things we couldn't before.



Confidential

© 2026 Mindera

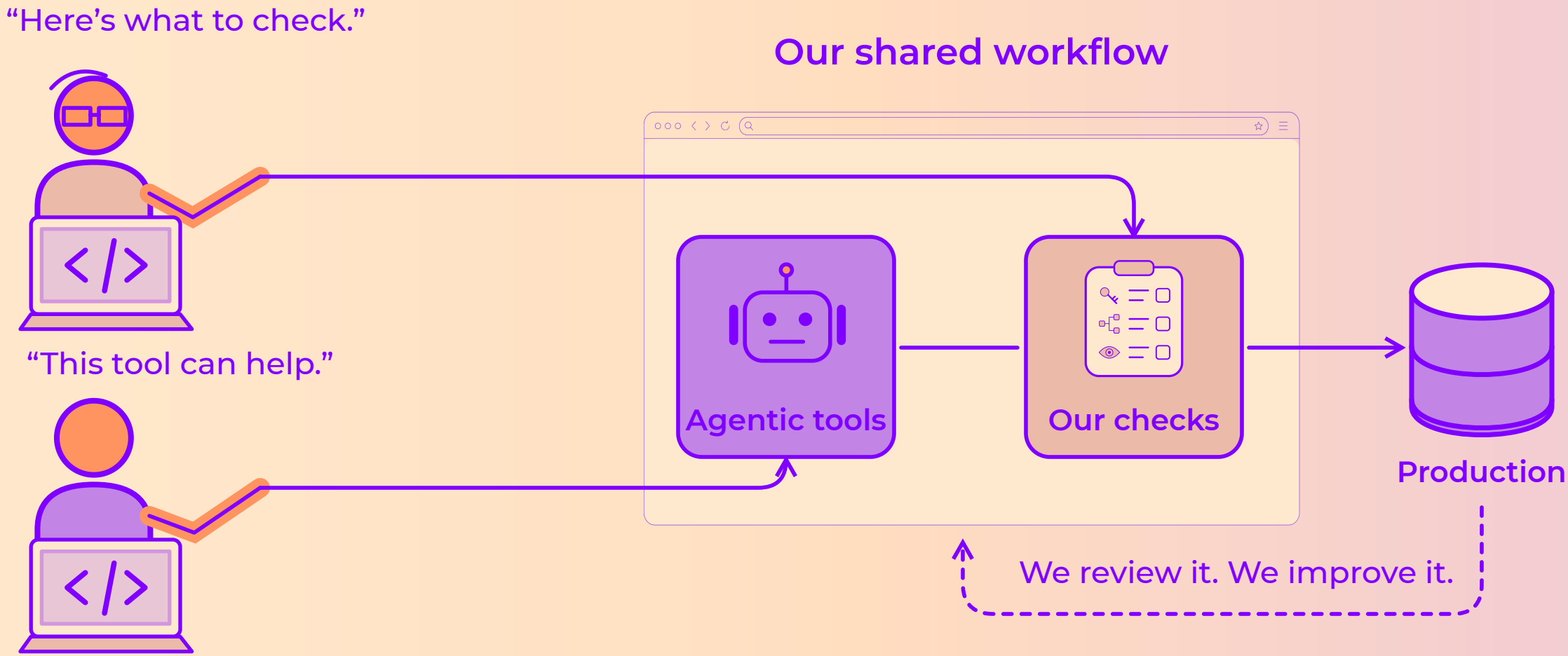
Bring business colleagues into the work.



Confidential

© 2026 Mindera

Now we can get better together.



Confidential
© 2026 Mindera

Thank you.

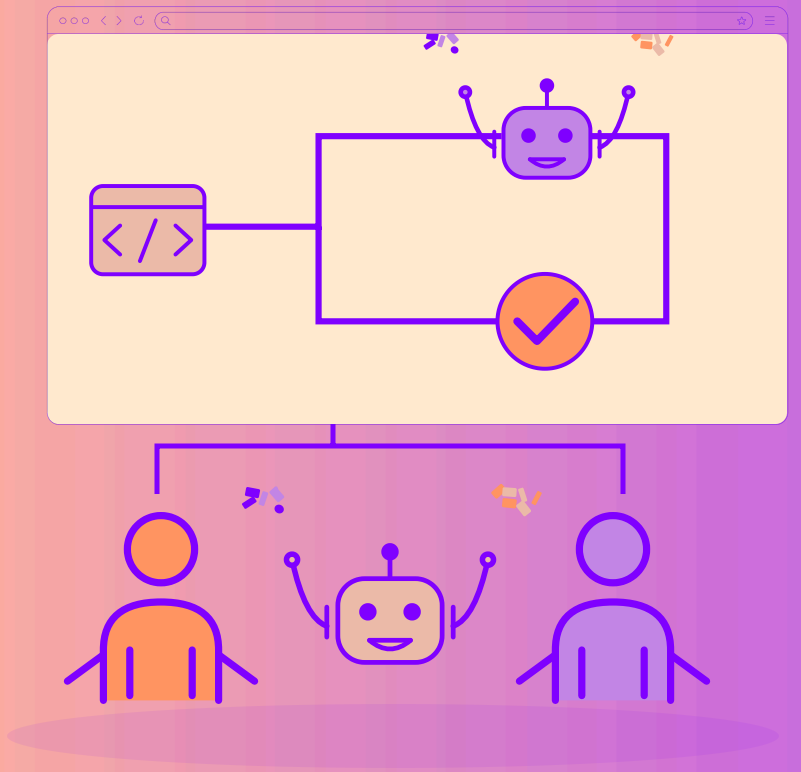
Where would your team start?

Questions?

João Anes

(pronounced joe-WOW)

joao.anes@mindera.com



Confidential

© 2026 Mindera