

Lesson 4: Trust but Verify

I knew the phrase *“trust but verify”* long before I learned how to practise it. That took several production rollbacks, numerous 6 a.m. meetings, and busy weekends spent in lengthy Slack huddles.

The pattern kept repeating.

Engineering would report that the work was done. I would take that update at face value and communicate the same confidence upwards. Then stakeholders would use the product and experience something different from what they expected.

Other times, QA would confirm that the flows had been tested and the tickets approved in staging. I would instruct engineering to move the work into production. Something would fail. We would roll back.

What had appeared to be completed work became another investigation, another difficult conversation, and another interruption to everyone’s plans.

The wrong lesson would have been to stop trusting the team.

The lesson I needed was more uncomfortable because it involved my own judgement. I had accepted an update without understanding its boundaries. Then I had turned that update into a broader commitment. I had mistaken a status update for evidence of readiness.

“Done” was not self-explanatory. Staging approval answered some questions, but it did not automatically answer every question implied by a production release.

Before passing confidence upwards, I needed to understand what had been tested, under which conditions, what remained uncertain, and what we were actually ready to promise.

That did not require me to repeat everyone’s work. It required me to stop filling the gaps with assumptions. Verification would not have made every release flawless. It would have made those gaps visible before we committed around them.

The Idea

Trust but verify means working from good faith while checking the claims that influence important decisions. You trust people to bring their expertise, do their work, and tell you what they know. Then you verify what has been established before using it to justify a promise, a priority, or a release.

This is not about making the product manager the organisation's final authority on truth. It is about giving everyone a shared understanding of the evidence, its limitations, and the risk being accepted.

A customer request is not automatically a priority. A successful test is not proof that every supported scenario works. A confident prediction is not a guaranteed outcome.

Each may be useful. None should silently become more than it is. Trust the people. Verify the claim. Make the assumptions visible.

Do This Next: Four Verification Habits

1. Name the Claim and the Decision

Start with two questions:

- What is being asserted?
- What decision depends on it?

"Users hate onboarding" is not yet a useful problem statement.

Which users? What did they say or do? Where did they struggle? What evidence suggests why? Do not turn a vague claim into a precise but invented explanation.

"Users struggle with onboarding" does not automatically mean "users drop off at step three because the instructions are unclear."

The behaviour, the location, and the proposed cause are separate things to investigate before establishing what is at stake.

Are we considering a redesign? Changing acquisition spend? Delaying another feature? Increasing onboarding support? The decision tells you which uncertainty matters.

Without this connection, verification can become interesting research that never changes what the team does.

2. Match the Check to the Consequences

Verification should reflect the consequences of being wrong, not simply the amount of product (design, engineering, qa, e.tc) work involved.

Ask:

- What happens if the claim is wrong?
- Who would be affected?
- How difficult would the decision be to reverse?
- What happens if we wait?
- What work would we displace by acting now?

Treat urgency as something to understand, not something to accept or dismiss automatically. If nobody can explain the consequence of waiting, the urgency remains unverified. That does not prove there is no risk. It means the risk needs clarification.

When credible, immediate harm is possible, involve the appropriate owner promptly. Protective action and verification may need to happen together. The objective is not perfect certainty. It is enough confidence to make a responsible decision.

3. Choose the Check That Can Answer the Question

There is no mandatory sequence for verifying every claim. Choose the lowest-effort, reliable check that could change the decision.

Useful routes can include:

- **Direct observation:** Use the product or observe the workflow.
- **Original material:** Review the customer message, ticket, recording, log, specification, or test result.

- **Customer evidence:** Understand what customers say, what they do, and the circumstances surrounding both.
- **Data analysis:** Check patterns, segments, trends, and measurement definitions.
- **Independent perspectives:** Ask people with different knowledge or access to evidence what they have observed.

A customer email may clarify a purchasing requirement faster than product exploration. Logs may be more useful than clicking through an interface when investigating an intermittent failure.

For metrics, establish what is being measured before interpreting the movement. *“Retention is down”* needs a definition, a segment, a comparison period, and confidence that the measurement has not changed.

For bugs, make a focused attempt to reproduce the issue where appropriate. Use approved environments, permitted data, and agreed safeguards. Any production checks need coordination with the responsible owners.

If you cannot reproduce the problem, record that accurately. One important caveat, though: *“not reproduced” does not mean “not real.”*

Also check whether your sources are genuinely independent.

Sales and Support may both be repeating the same customer story. Two roles do not automatically mean two sources. The same applies to AI-generated material. A transcript, an AI recap, and a presentation based on that recap are still drawing from the same underlying conversation.

Trace important claims back to their sources. Distinguish what the source establishes from what someone, or something, has inferred.

4. Decide and Communicate the Boundaries

Verification should always end in a decision or a specific next check. It should not become an open-ended search for reassurance. So, record what the evidence supports, what remains uncertain, and what you are choosing to do.

For customer requests, frequency and severity are useful inputs, not automatic priority labels. Consider the affected customers, strategic relevance, alternatives, consequences, and opportunity cost.

For delivery commitments, establish what “done” includes. Implementation complete, agreed tests passed, available in staging, and ready for a particular release are related states. They are not interchangeable.

For any meaningful commitment, make the trade-off explicit:

“We are choosing A over B because of X. We still do not know Y. We are accepting that risk, and we will revisit the decision if Z happens. Are we okay with this?”

If more information is needed, identify the smallest useful check, its owner, and when the decision will return. “Not yet” is a responsible answer when it comes with a clear path forward.

Worked Example: “SSO Will Win This Deal”

The following example is illustrative.

The CEO wants single sign-on (SSO) added to the roadmap. Sales believes an enterprise prospect will buy if the team commits to delivering it.

The request arrives as a confident statement:

“If we build SSO, this deal is ours.”

Engineering has not yet assessed the exact requirement. Accepting the request immediately would displace already existing priorities.

Here is what a completed Verification Brief might reveal.

1. Claim and Source

Claim: Delivering SSO will secure the enterprise deal.

Source: Sales, based on a conversation with the buyer.

The founder has heard the same account from Sales. That is repeated information, not independent confirmation.

2. Decision

Should we displace current work and promise an SSO delivery date?

3. Consequences and Urgency

If we promise prematurely, we may disrupt committed work and create a deadline the team cannot meet.

If we decline without investigating, we may lose a worthwhile opportunity.

The customer wants an answer, but wanting an answer is not the same as having an agreed implementation deadline.

4. Evidence

The customer's security checklist confirms that SSO is a purchasing requirement.

A follow-up conversation establishes that the customer needs support for one specified identity provider. Automated user provisioning is not required for this purchase. However, pricing and procurement remain unresolved.

Engineering has not yet estimated the agreed scope. The evidence supports a narrower conclusion than the original claim: SSO is necessary for this purchase. It is not sufficient to guarantee the purchase.

5. Remaining Uncertainty

The decision depends on two unresolved questions:

- What would it take to deliver the customer's actual requirement?

- Are the other purchasing conditions likely to be satisfied?

The next useful checks are an engineering assessment of the written requirement and confirmation of the outstanding buying conditions.

6. Decision and Trade-Off

Do not promise a delivery date or treat the revenue as secured. Authorise a bounded assessment of the requirement before changing implementation priorities.

Once the assessment is available, compare the opportunity with the work it would displace. Any customer commitment must reflect the agreed scope and the team's actual confidence.

The trade-off is explicit: the company accepts the possibility that the buyer may reject a conditional answer rather than making an unsupported promise.

7. Follow-Through

Sales confirms the written requirement and remaining purchasing conditions.

Engineering assesses scope, dependencies, and delivery uncertainty.

The product manager brings those findings back to the relevant decision-makers.

Revisit the decision if the requirement expands, the estimate changes feasibility, or another purchasing obstacle makes the opportunity less likely.

What Changed?

The team began with a binary choice:

Build SSO or lose the deal.

Verification produced a more useful picture:

1. The requirement was real.
2. The scope was narrower than it first appeared.

3. The sale was still uncertain.
4. The delivery commitment was not yet supported.
5. The next action had an owner and a purpose.

Nobody needed to accuse Sales of exaggerating or dismiss the founder's commercial instinct. The team needed to separate a verified requirement from an unverified prediction.

That is what makes verification useful.

Say This: Language for Shared Confidence

When Someone Makes a Strong Claim

"What do we know directly, and what are we inferring?"

"Can we look at the original example behind that?"

When You Need to Pass an Update Upwards

"Before I communicate this, can we clarify exactly what is complete and what remains uncertain?"

In the situations that led to my rollbacks, this was the distinction I needed to preserve.

Instead of: *"The feature is done."*

A more accurate update would have been:

"The agreed scenarios have passed in staging. Production rollout is still pending. These checks and limitations remain outstanding."

The second statement gives stakeholders something they can act on without implying more than the evidence supports.

With Engineering and QA

“Can we walk through what the approval covers so I communicate it accurately?”

“What do you need from product to help us assess readiness?”

When a Request Is Urgent

“What changes if we act now rather than later?”

“If we bring this forward, what should move out?”

When You Need to Expose Your Own Assumption

“I am assuming X because of Y. That may be wrong. What is the most useful way to check it?”

Use these questions during ordinary collaboration, not only when you disagree with someone.

Watch Out for These Failure Modes

Failure Mode 1: Verifying Everything

If every small decision requires an investigation, the process may cost more than the uncertainty it removes.

Correction: Match the effort to the consequences and reversibility of the decision. Do not impose the same evidence requirements on every kind of work.

Failure Mode 2: Turning Verification into Policing

You can ask a reasonable question in a way that makes a colleague feel publicly examined. You can also undermine specialist ownership by treating your personal check as more important than the team's testing.

Correction: Agree what evidence matters with the relevant owners. Contribute customer context and decision clarity. Do not substitute yourself for engineering, QA, or other specialists.

Failure Mode 3: Hiding Behind Uncertainty

More information does not always produce a better decision. Sometimes it only delays an uncomfortable one.

Correction: Identify the uncertainty most likely to change the decision. Timebox the useful checks, then choose with the best available evidence. Record the remaining risk and what would trigger reconsideration.

Failure Mode 4: Verifying Selectively

A claim that supports your bias receives a quick nod. A claim that threatens it receives an interrogation.

That is not rigour. It is confirmation bias wearing a space coat.

Correction: Apply comparable standards to claims with comparable consequences. Make your own assumptions visible, seek contradictory evidence, and explain what would change your mind.

Tool: The Verification Brief

Use this when a claim could materially change a priority, promise, or release. Keep the summary to one page and link to supporting material.

1. Claim and Source

What is being asserted? Where did the information originate?

2. Decision

What decision depends on this claim?

3. Consequences and Urgency

What happens if we are wrong? What happens if we wait? How reversible is the decision?

4. Evidence

What supports the claim? What contradicts it? Are the sources independent? What does the evidence actually establish?

5. Remaining Uncertainty

What could change the decision? What is the smallest useful check?

6. Decision and Trade-Off

What are we choosing? What moves out? What risk remains, and who owns what?

7. Follow-Through

Who does what next, by when? What signal or event would make us revisit the decision?

Closing

Those rollbacks and lengthy meetings with higher-ups did not make me trust my team less. They taught me to pass on confidence more carefully.

A useful question belongs between an update and a commitment: ***What has been verified, and is it enough for the decision we are about to make?***

You will not prevent every failure by asking it, but you give the team a better chance of discovering its assumptions before customers, stakeholders, or a production incident expose them.

That is the habit.

Next, we turn it towards the roadmap: moving from a list of requests to a set of reasoned bets.