XONIK

GUIDE

Technology Architecture Review Checklist

A checklist for reviewing a technology architecture: what depends on what, how change reaches production, how it recovers, who owns it and what leaving would cost.

Version	1.0
Date	20 October 2026
Published by	XONIK
For	A CTO or architect preparing or commissioning an architecture review.
Use with	The Technology and Architecture Risk Assessment on the XONIK website, and the Technical Debt Register.

How to use this checklist

Use it with the people who run the systems as well as the people who designed them. For each statement, ask to be shown the evidence. A statement is true when someone other than its author can demonstrate it today. Write what you find on the review record at the end, with an owner for each finding.

The checklist asks questions and tests nothing. It is not a security assessment or a penetration test.

Dependencies and boundaries

- [ ] A current map shows what each major system depends on and what depends on it, and the map has an owner.
- [ ] The boundaries between parts are written down: what crosses each one and what each side may assume.
- [ ] Dependencies on outside suppliers that would stop the business if they failed or changed terms are listed, with a plan for each.
- [ ] No part reaches into another part's data store directly.
- [ ] Each part has one team that answers for it.

Questions to ask

- Which change last month surprised someone because of a dependency nobody knew about?
- What happens to the business if each outside supplier disappears for a week?

Data and integration

- [ ] Each kind of data has one system of record, and the rest are copies.
- [ ] Where systems disagree, there is a rule for which one wins, and a way to find the differences.
- [ ] Integrations are documented with their owners, their direction and what happens when one side is down.
- [ ] Data that must be kept, and data that must be deleted, are identified and the rule is applied.
- [ ] Reports and decisions rely on a named source instead of a copy someone made.

Questions to ask

- If a customer record differs between two systems, who finds out, and how?
- Which integration would nobody know how to repair?

Change and release

- [ ] A change reaches production by the same steps every time.
- [ ] Changes are tested before release, and a release can be reversed.
- [ ] Small changes do not need a meeting to coordinate.
- [ ] Emergency changes follow a route, and each is reviewed afterwards.
- [ ] The environments used for testing match production closely enough to trust the result.

Questions to ask

- What stops a team from releasing a small change this afternoon?
- When did a release last have to be reversed, and how long did that take?

Reliability and recovery

- [ ] A critical system has been restored from backup, into a clean environment, to prove that it works.
- [ ] The recovery steps are written down and someone other than their author has followed them.
- [ ] If the person who knows a critical system were unavailable, someone else could run it.
- [ ] Alerts tied to what matters to users catch failures before customers do.
- [ ] Alerts are tuned so that people still trust them.

Questions to ask

- What is the longest the business could be without each critical system, and does the recovery plan fit it?
- Which system has never been restored from backup?

Security basics

- [ ] Access to systems is by person and by role, and leavers lose it promptly.
- [ ] Secrets live in a managed place, and none sit in code or in messages.
- [ ] Software and dependencies are kept up to date by a routine.
- [ ] Logs show who did what to which system, and someone can read them.
- [ ] Anything a regulator or a customer expects in this area is identified and has an owner.

Questions to ask

- Who can change production without a second person seeing it?
- What would you want logged after an incident that is not logged today?

Ownership and knowledge

- [ ] Architecture decisions, and the reasons for them, are written where new engineers can find them.
- [ ] Each system has a named owner with the authority to change it.
- [ ] Knowledge of how systems connect is documented and shared, and no single person holds it.
- [ ] A new engineer can run the system locally and make a small change in their first days.
- [ ] Technical debt is recorded, ranked and reviewed on a schedule.

Questions to ask

- Which system would nobody dare to change?
- If one engineer left tomorrow, what would stop?

Cost and exit

- [ ] The running cost of each major system is known, including usage charges and the time spent keeping it working.
- [ ] For each major supplier, the cost and effort of leaving have been estimated.
- [ ] Data can be exported in a form another system could read.
- [ ] Contracts do not lock in terms that the architecture cannot meet.
- [ ] Spending is reviewed against use on a schedule.

Questions to ask

- What would it take to move off your largest supplier?
- What do you pay for that nobody uses?

Review record

Write one line for each statement you could not tick. Rate each finding as low, medium or high by what it would cost if it went wrong, and say who owns the answer.

Finding	Cost if it goes wrong (low, medium, high)	Owner	Decision and date

Limits of this document

XONIK wrote this document as a working aid. It is general guidance, not legal, financial, technical-audit or regulatory advice, and it certifies nothing. It records your own reading of your own organization, so it is only as good as the answers you give. It does not guarantee any result. A completed review records what you were shown, which may not hold for every system.