

COMPOSABLE SCHEMAS

One schema file is fine. Until three teams own it.

A single schema works beautifully right up to the point where the documents team, the billing team and the platform team all need to change it in the same sprint. Then it stops being a model and becomes a merge conflict with a review queue attached.

How the bottleneck forms

It is rarely a design failure. One file is the right answer for one team, and it stays the right answer for a surprisingly long time. The trouble starts when the file outgrows the group of people who understand all of it.

At that point every change needs a reviewer with context on definitions they do not own, two unrelated changes land in the same file in the same week, and a shape that half the objects need — ownership, audit access, tenancy — gets copied rather than shared, because copying is the only move that does not require a conversation.

What composition changes

Split the schema into files with owners. Express the shapes that recur as **partials**. Import them where they are needed, and assemble the whole thing at deploy time.

The important property is what does *not* change: SpiceDB still serves **one assembled schema**. The runtime, the checks and the consistency model are identical. This is a source-organisation feature, not a distributed one — which is exactly why it is safe to adopt gradually.

The review that matters still happens. It happens **on the shared file** — which is where you wanted the scrutiny concentrated in the first place, rather than spread evenly across every change anyone makes.

CONTENTS

1. Three files, three owners
2. What actually gets deployed
3. The shape written once
4. Ownership without a bottleneck
5. What to watch for
6. Adopting it incrementally

AUDIENCE

Platform and security engineers whose schema is shared by more than one team

NOTE ON SYNTAX

Illustrative of the feature. Take the specification from authzed.com/docs

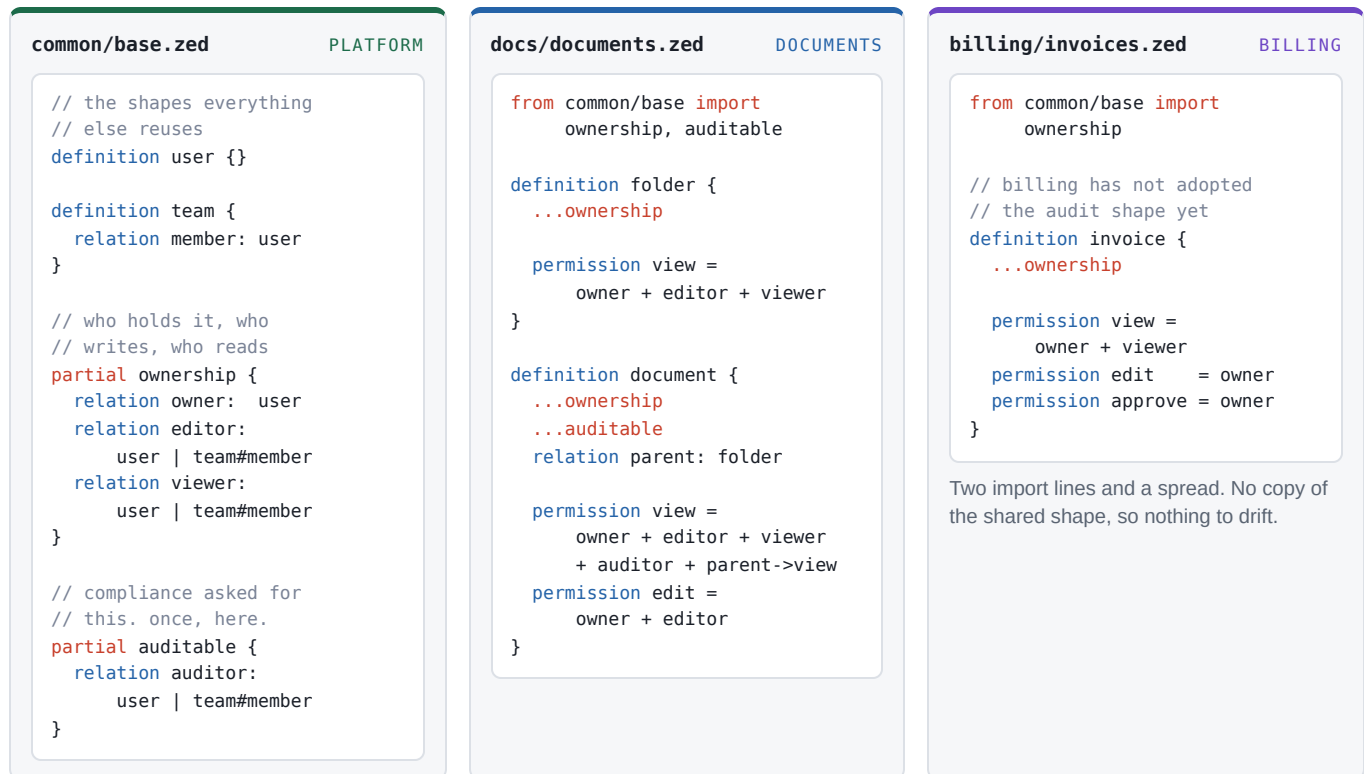
READING TIME

About eight minutes

1 · THREE FILES, THREE OWNERS

Each file has a team on it

The platform team owns the common shapes. The documents team and the billing team own their own objects. Neither needs the other to review a change.



2 · WHAT ACTUALLY GETS DEPLOYED

Composition happens before deployment

The composer resolves the imports and expands the partials. What SpiceDB receives is a single schema, exactly as if it had been written by hand.

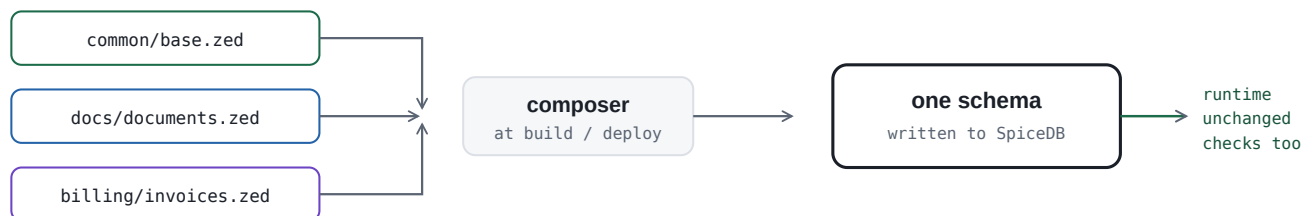


Figure 1 – Nothing about the serving path changes. The split exists in your repository, not in the system that answers checks.

3 · THE SHAPE WRITTEN ONCE

A worked example, in four moves

Compliance asks that auditors be able to read anything subject to review. The platform team writes that shape once. Here is what happens next.

MOVE	WHAT HAPPENS
The shape is written	<code>partial auditable</code> lands in the platform team's file. One relation, one place, one owner.
Documents adopts it	Two lines in their own file: the import, and <code>...auditable</code> inside the definition. Grace, on the audit team, can now read a document — through a relation the documents team never wrote themselves.
Billing has not	Same person, same audit team, no route to an invoice . Adoption is per file, so the shape spreads at each team's pace rather than on a flag day.
Billing adopts it	They import the partial and spread it in. One line, inside their own file, no platform review, and no second copy of the audit shape to keep in step with the first.

The thing that was expensive before was not writing the audit relation. It was writing it **eleven times**, in definitions owned by five teams, and then keeping those eleven copies identical through two years of unrelated changes.

4 · OWNERSHIP WITHOUT A BOTTLENECK

Three things this changes about how teams work

REVIEW LANDS WHERE IT BELONGS

The billing team changes billing's file and ships. A platform engineer no longer reviews diffs they have no context for. The **shared file** still gets scrutiny, and now it gets more of it, because it is no longer buried among routine changes.

COMMON SHAPES STOP BEING COPIED

Ownership, audit access, tenancy, soft-delete — the same handful recur across nearly every object a platform has. When compliance changes what auditors may reach, it changes **in one place** rather than in a set of definitions that were supposed to match.

ADOPTION IS PER FILE

A new shared shape does not require every team to move at once. Each file imports it when that team is ready, and the composed schema is valid at every point in between. There is no half-migrated state to manage.

THE PROPERTY THAT MAKES THIS SAFE

Composition is a **build-time** concern. There is no runtime resolution, no cross-file lookup on the request path, and no new failure mode in the serving layer. If the composer produces a valid schema, SpiceDB behaves exactly as it would have with the same schema typed into one file — which also means your existing tests remain meaningful.

5 · WHAT TO WATCH FOR

Three things it does not fix

IT DOES NOT MAKE THE MODEL SIMPLER

The composed schema is exactly as complex as the one you would have written by hand. If a schema is hard to reason about, splitting it across files will not fix that — and it can hide it, by putting the pieces of a confusing permission in three places instead of one.

Read the composed output. Make it an artifact in your build that a human can open, not an intermediate nobody ever sees.

A SHARED PARTIAL IS A SHARED DEPENDENCY

Changing one changes every definition that imports it. That is the point, and it is also the risk — a one-line change to [ownership](#) touches every object on the platform.

Treat the common file as you would a shared library: **fewer owners, more review, and a test that exercises the composed result** rather than each file alone.

AND ONE ON SYNTAX

The [partial](#) and [from ... import](#) forms in this paper illustrate the idea rather than pinning the specification. **Composable schemas have moved as the feature matured**, so take the current syntax, tooling and constraints from [authzed.com/docs](#) before writing anything you intend to keep. Nothing in the argument above depends on the exact spelling.

6 · ADOPTING IT INCREMENTALLY

You do not have to reorganise anything to start

STEP	WHAT TO DO
Leave it alone	If one team owns the schema and changes land without contention, stop here. Composition solves a coordination problem you do not have, and the single file is easier to read.
Split by owner first	Before introducing any partials, move each team's definitions into their own file. This alone removes most of the merge pressure, and it is reversible.
Promote one shape	Find the relation set you have already copied two or three times — it is usually ownership. Make it a partial, and have one team adopt it. Check the composed output against the schema you had before.
Then the rest	Each remaining team adopts when a change brings them past it. There is no deadline, because the composed schema is valid throughout.

The test for whether you need it

How many people must approve a schema change today, and how many of them understand the part being changed?

If those two numbers are the same, you have no problem to solve. When the first is larger than the second, the review has stopped being a safeguard and started being a queue — and that gap is what composition closes.

SAID PLAINLY

This is an organisational feature, not a performance or capability one. It does not make checks faster, does not change what you can express, and does not reduce the complexity of your model. What it changes is who can ship a schema change without waiting for someone else — which matters a great deal at four teams and not at all at one.

Try it: the SpiceDB Playground runs in a browser with nothing to install. Source at github.com/authzed/spicedb.