WHITEPAPER · 2026

Scaling Power BI to external users

Power BI does not scale. Not the way you're sharing it.

DataTako
Licensing, capacity, cost & control
in the Microsoft Fabric era

From chaos to scalable
Power BI delivery

WHAT'S INSIDE

The guide for everyone sharing Power BI outside their own walls.

Embedding Power BI for users and partners is easy to start and hard to scale. Licensing traps, capacity throttling and runaway cost rarely show up in the demo — they show up at customer #40. This paper maps the decisions that matter, with the 2026 Fabric reality baked in.

01 · THE PROBLEM

Why Power BI breaks the moment you share it externally.

Power BI was built to make internal reporting effortless. Connect, model, publish, share inside your tenant. It is brilliant at exactly that — and that is also the trap. The model that works for fifty colleagues quietly falls apart the moment your audience becomes **external**: users, partners, franchisees, the public.

The cracks are predictable:

- **Licensing.** Every interactive viewer "needs a Pro license" — until your user count makes that absurd, both in cost and in administration.
- **Access management.** Inviting external users into your tenant, juggling guest accounts and workspace permissions, becomes a security review waiting to happen.
- **Tenant boundaries.** Your data and your customers' identities living in one tenant is fine at five customers and frightening at five hundred.
- **Manual delivery.** "Just export it to PDF and email it" is not a delivery strategy. It is technical debt with a monthly cadence.

Sharing Power BI internally is a feature.
Sharing it externally is an architecture decision.

Treat external delivery as a product, not as a setting you flip. The teams that scale cleanly decide this on purpose — early.

None of this means Power BI is the wrong tool. It means the **delivery model** around it is where scale is won or lost. The rest of this paper is about that model.

02 · THE BOTTLENECK

It's not the features. It's licensing and capacity.

Ask why a Power BI rollout stalled and you'll rarely hear "we ran out of visuals." You'll hear "the licensing got out of hand" or "the reports got slow under load." Two levers decide whether external sharing scales — and neither is on the report canvas.

Lever 1 — Who needs a license

In a normal tenant, anyone who interacts with a report needs a per-user **Pro** or **Premium Per User** license. Multiply that by an external audience and the math collapses. The escape hatch is capacity-based licensing and, crucially, the **embed-for-your-customers** model — where your application authenticates, and end users need **no Power BI license at all**.

Lever 2 — Whether the engine keeps up

Capacity is a fixed pool of compute (measured in Capacity Units). Refreshes, queries and rendering all draw from it. Exceed it and Power BI **throttles** — reports slow, then stall. Scaling externally without capacity planning is the single most common reason embedded analytics "feels broken" in production.

0
Power BI licenses needed for external users in the embed-for-your-customers model

CU
Capacity Units are the currency — every refresh and query spends them

F64
The Power BI Service threshold for license-free internal viewing — App Owns Data skips it

THE REFRAME

Stop asking "how many licenses do we need?" Start asking "which capacity model fits our audience, and how do we keep it from throttling?" That question scales. The license question doesn't.

The Fabric-era capacity map, in plain language.

This is where most older guides are out of date. They still treat **P-SKUs** (classic Premium capacity) as the default. Microsoft has steered Premium onto **Microsoft Fabric F-SKUs**, and new P-SKU commitments are no longer the path forward. If a "definitive guide" doesn't mention Fabric, it's describing 2022.

Capacity	What it is	Best for	Cost behaviour
A-SKUs (A1–A6)	Azure pay-as-you-go embedded capacity, billed hourly	Embedding to external customers; dev/test; spiky or seasonal load	**Pausable** — stop the clock when idle
F-SKUs (F2–F2048)	Microsoft Fabric capacity — the current strategic platform	New deployments wanting Fabric features and a future-proof path	Pausable; cheaper when reserved annually
P-SKUs (P1–P5)	Legacy Power BI Premium capacity	Existing estates only — being transitioned to Fabric	Annual commitment; not the path for new builds
EM-SKUs	Older embed SKUs, monthly commitment	Legacy — avoid for new projects	Fixed monthly

The F64 line — and why App Owns Data sidesteps it

The F64 rule only applies to one path: internal users consuming reports directly in the Power BI Service. At **F64 and above**, those viewers don't each need a Pro license; below F64 they do. That's the threshold everyone quotes — and then misapplies.

With App Owns Data, F64 is irrelevant — for internal *and* external users.

When your application owns the data and authenticates on the viewer's behalf, nobody viewing through your app needs a Power BI license — at any SKU size, F2 included. App Owns Data isn't only for external customers: embed it for your own staff too and the F64 question disappears entirely. F64 only bites if people open reports in the Power BI Service instead of through your app.

PRACTICAL TAKEAWAY

For external customer portals, size capacity to **load**, not to the F64 license rule. Use pause/scale to match real usage — start smaller than you think, monitor, and scale on evidence.

F-SKUs, A-SKUs, OneLake and the end of Premium.

The capacity map above is enough to start. This page is for the decision you'll actually agonise over: **A-SKU or F-SKU**, and what Fabric changes for external delivery.

F-SKUs vs A-SKUs — both can embed

A-SKUs (Power BI Embedded)	F-SKUs (Microsoft Fabric)
Pure Azure pay-as-you-go, billed per hour	The full Fabric platform: lakehouse, pipelines, OneLake, Power BI
Pause/resume in seconds — ideal for spiky or business-hours load	Pausable too, plus a reservation discount for steady load
Simplest billing for a pure embedding use case	F64+ also unlocks license-free internal viewing
No broader Fabric workloads	Future-proof — this is where Microsoft is investing

When to choose which

- **Choose an A-SKU** when embedding to customers is your only goal, billing simplicity matters, and you want to pause aggressively to cut cost.
- **Choose an F-SKU** when you also want Fabric workloads (data engineering, pipelines, OneLake), need the F64 internal benefit, or want the strategic, future-proof path.

How OneLake changes external delivery

OneLake is Fabric's single, logical data lake for the whole tenant. Two things matter for embedding: **shortcuts** let you reference data in place without copying it across customers, and **Direct Lake** mode serves reports at near-import speed straight off the lake — no scheduled refresh to break, fewer moving parts in a multi-tenant pipeline.

> **THE FUTURE OF P-SKUS**
>
> Treat classic Premium (P-SKUs) as legacy. New commitments go to Fabric F-SKUs, and existing P estates are on a transition path. If you're architecting external delivery in 2026, build on F (or A) — and plan any P migration deliberately, not under deadline pressure.

Three ways to share externally. One wins for scale.

Method 1 — Invite as guest users

Add external people to your tenant via Entra ID B2B and assign licenses. **Fine for a handful of trusted partners.** It does not scale: every user is an identity you manage, a license you buy, and a guest account in your security perimeter.

Method 2 — Publish to web

The "publish to web" embed code makes a report **public to anyone with the link**. No security, no row-level filtering, fully indexable. Useful for genuinely public data only. For customer data it is a breach waiting to happen — never use it for anything confidential.

Method 3 — Embed for your customers ("App Owns Data")

Your application authenticates to Power BI using a **service principal**, generates short-lived **embed tokens**, and serves reports inside your own UI. End users log into your app — they never touch Power BI, never need a license, never see your tenant.

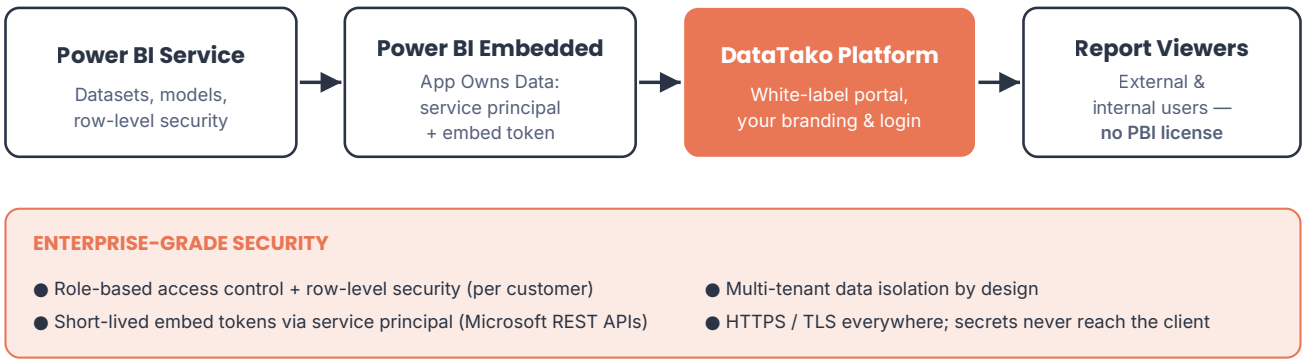

Fig. 1 — The App Owns Data architecture: your app authenticates, your customers just view.

Why App Owns Data wins

- ✓ Zero Power BI licenses for end users
- ✓ Your branding, your login, your UX
- ✓ Row-level security scoped per customer
- ✓ Tenant and customer identities stay separate
- ✓ Scales to thousands of external users

What it asks of you

- – Service-principal setup and token logic
- – A capacity sized and monitored for load
- – Secure token generation in your backend
- – Multi-tenant data isolation by design
- – Ongoing maintenance as Power BI evolves

How to size capacity, step by step.

Capacity sizing is where teams either overpay from day one or get throttled in week two. You don't guess your way to the right SKU — you start small, measure, and scale on evidence. Here is the loop.

1. **Count your audience and concurrency.** Start from total external users, then estimate realistic peak concurrency (often 5–15%, not 100%). Fifty concurrent sessions is a very different machine than five hundred named users.
2. **Profile the report load.** Model size, visual complexity, import vs DirectQuery, and refresh frequency all spend Capacity Units. A heavy DirectQuery page costs far more per render than a lean import model.
3. **Start one SKU smaller than instinct says.** Begin at an entry SKU (e.g. F2–F8 or A1). You can scale up in minutes, so there's no reason to pre-buy headroom you can't yet justify.
4. **Load-test before launch.** Simulate concurrent renders and refreshes at expected peak. Find the throttling point in a test, not in front of a customer.
5. **Monitor with the Capacity Metrics app.** Watch CU utilisation, interactive vs background operations, and any throttling. This is your single source of truth — not a hunch.
6. **Scale (or autoscale) on evidence.** Move up a SKU only when the metrics justify it. Sustained high utilisation = scale up; quiet evenings = pause or scale down.
7. **Pause and schedule for idle windows.** Business-hours-only portals don't need to pay 24/7. Pausing an A-SKU overnight and at weekends is one of the largest single cost levers available.
8. **Re-evaluate quarterly.** Usage grows, reports get heavier, customers get added. Once load is predictable, switch to a reservation for the discount.

Model your own numbers

Plug your user count, concurrency and SKU into the DataTako cost calculator for a live estimate: datatako.com/cost-calculator

07 · THE COST MODEL

What actually drives your bill.

Embedded cost is a stack, not one number — and most surprises come from one layer being mis-sized. Here is the whole stack, and the levers that move it.

Cost layer	What it is	The lever
Capacity (A/F-SKU)	The dominant, recurring cost. Scales with the SKU you pick.	Right-size to load; pause when idle; reserve annually
Author / dev licenses	Pro licenses for whoever builds and publishes reports	Small, fixed team — not per end user
App hosting (Azure)	The web app that embeds the reports	Standard cloud right-sizing
Data services	Databases, lakehouse, pipelines feeding the reports	Optimise models; incremental refresh
Build & maintenance	Engineering time to build and keep it running	Build-vs-buy — see section 09

THE BIGGEST LEVER, IN ONE LINE

Pay-as-you-go capacity you **pause** outside business hours, or **reserve** annually for steady load, is where the savings live. Reserved is materially cheaper than on-demand — if your usage is predictable enough to commit.

Illustrative monthly shape (not a quote)

Below is the shape of a small external-portal deployment — **directional, not prices**.

Component	Driver	Relative weight
Embedded capacity (entry A/F-SKU)	Single SKU, paused nights/weekends	●●●● dominant
2–3 Pro author licenses	Your report team	● small, fixed
Managed platform & hosting	Run by DataTako, not your team	●● subscription
End-user licenses	None — App Owns Data	— zero

Microsoft pricing changes frequently and varies by region and commitment. Always verify against the DataTako cost calculator (datatako.com/cost-calculator), the Azure Pricing Calculator, and the current Power BI Embedded / Microsoft Fabric pricing pages before budgeting.

08 · WORKED EXAMPLE

500 external users, end to end.

Abstract advice is easy to nod at and hard to apply. So let's size a real-shaped scenario: a SaaS vendor giving **500 external users** access to embedded dashboards.

> **THE ASSUMPTIONS**
>
> 500 named external users · realistic peak concurrency ~10% (≈ 50 concurrent sessions) · moderate report complexity, import mode, one daily refresh · business-hours usage, quiet at night and weekends.

How it plays out

- **Delivery model:** App Owns Data. The vendor's app authenticates with a service principal; users log into the vendor's portal, not Power BI.
- **End-user licensing:** zero. None of the 500 need a Power BI license — that's the whole point of embed-for-your-customers.
- **Capacity:** start at an entry SKU, load-test 50 concurrent renders, and scale up only if the Capacity Metrics app shows sustained pressure. Pause overnight and weekends.
- **Author licenses:** 2–3 Pro seats for the people who build the reports. Fixed, regardless of user count.

The contrast that sells the model

Per-user licensing (the naïve path)	App Owns Data (capacity)
500 × Power BI Pro, every month — plus inviting 500 guests into your tenant	One capacity SKU you can pause + 2–3 author seats
Cost scales **linearly** with every new user	Cost scales with **load**, not headcount
500 identities to manage and offboard	Identities stay in your app; tenant stays clean

Your 500 may not be our 500

Concurrency and report weight change everything. Run your exact numbers: datatako.com/cost-calculator

09 · BUILD VS. BUY

Build it yourself, or buy a platform?

Both are legitimate. The honest answer depends on your engineering capacity and how core embedded analytics is to your roadmap. Here's the trade-off without the sales gloss.

Build your own

STRENGTHS

Total control over UX, auth and data handling. Fits unusual requirements. No vendor dependency.

COSTS

Real Power BI + dev expertise required. Months to production. Token logic, multi-tenant isolation and capacity monitoring are yours to own — forever.

Buy a platform

STRENGTHS

Live in days, not months. White-label and multi-tenant out of the box. Vendor carries maintenance and Power BI changes.

COSTS

Subscription fee. Less freedom than a bespoke build. You depend on the vendor's roadmap.

Most teams over-estimate the build and under-estimate the maintenance.

The first version is the cheap part. Keeping a custom stack secure and current — as Power BI, Fabric and Entra all evolve — is the cost that compounds. Price the second year, not just the first.

09 · THE COMPARISON

DIY vs Premium vs Embedded vs DataTako.

Four common routes to getting Power BI in front of external users, side by side. There is no universally "right" column — there is a right column for your situation.

Dimension	DIY custom build	Premium / F64+ sharing	Embedded (A-SKU, hand-rolled)	DataTako
External-user licensing	None (if built right)	Per-user Pro / guest access	None	None
White-label UX	Full, but you build it	**Limited** — Power BI UI	Full, but you build it	**Built in**
Multi-tenant isolation	You design it	Workspace-based, manual	You design it	**Built in**
Time to production	**Months**	Days–weeks	Weeks–months	**Days**
Maintenance burden	**High — all yours**	Medium	**High — all yours**	**Vendor-carried**
Cost model	Capacity + heavy dev	Capacity + per-user	Capacity + dev	Capacity + subscription
Best for	Analytics as core IP, strong team	Mostly-internal sharing	One-off embedding, in-house skills	Scalable external delivery, fast

"Premium / F64+ sharing" means giving users access through the Power BI service on a large capacity, rather than embedding. It's convenient internally but rarely the right fit for branded, external, multi-tenant delivery.

10 · THE HIDDEN COST

The year-2 maintenance reality.

"The first version is the cheap part." A custom embedded stack isn't a project you finish — it's a system you keep alive while Microsoft, your data and your customers all keep moving. This is the work that doesn't show up in the build quote.

Ongoing work	Why it never stops
Token logic updates	Auth flows, token lifetimes and refresh handling need maintenance as security guidance and SDKs change
API & SDK version changes	The Power BI REST and JavaScript APIs evolve; deprecations force code changes on Microsoft's timeline, not yours
Fabric migration	Moving from A/P-era setups onto Fabric F-SKUs and OneLake is real engineering, not a toggle
RLS drift	New customers, roles and data shapes mean row-level security rules need constant review or data leaks between tenants
Dataset refresh failures	Refreshes break on schema changes, source outages and credential expiry — someone has to catch and fix them
Capacity tuning	As load grows, SKUs need re-sizing, pausing schedules need adjusting, throttling needs chasing
Security patching	Dependencies, app hosting and the embedding layer all need ongoing patching and review

This is where DIY solutions quietly die.

Each item is survivable alone. Together, they're a standing commitment of engineering time that compounds every year — and it's precisely the burden a managed platform like DataTako absorbs for you.

Security: what *not* to do.

Most embedded security advice tells you what to implement. Just as useful is knowing the mistakes that show up again and again in real deployments — each one is a breach, a leak or an outage waiting to happen.

Anti-pattern	Why it bites	Do this instead
Master user for authentication	A single user account's credentials become a shared secret and a single point of failure	Authenticate with a **service principal**
Long-lived embed tokens	A leaked token stays valid for hours or days, far beyond the session that needed it	Generate **short-lived** tokens server-side, per session
Passing **tenant IDs / secrets** to the client	Anything in the browser is readable — secrets and identifiers leak to anyone who looks	Keep secrets server-side; never ship them to the front end
Publish to web for "temporary" user access	It's fully public and indexable — "temporary" public data has a way of staying public	Use App Owns Data, even for short-term access
Client-side filtering instead of **RLS**	Filters in the browser can be bypassed; the data is already there to be inspected	Enforce **row-level security** in the dataset

THE PATTERN BEHIND THE PATTERNS

Nearly every anti-pattern above is the same mistake wearing a different hat: **trusting the client**. Authentication, tokens, secrets and filtering all belong on the server. If a customer's browser could tamper with it, assume one eventually will.

12 · THE MATURITY MODEL

Where are you on the curve?

External Power BI delivery evolves through predictable stages. Find where you are — and notice that every level below the top carries a pain you've probably already felt.

L1	**Manual sharing.** Exporting to PDF/PowerPoint and emailing reports. Works for a few users; becomes a monthly chore that doesn't scale and has no security model.
L2	**Guest users.** External people invited into your tenant with licenses. Better, but every user is an identity, a license and a security-perimeter question.
L3	**Basic embedding.** A first App Owns Data integration for one product or customer. The pattern is right; multi-tenancy, monitoring and automation aren't solved yet.
L4	**Multi-tenant embedding.** Per-customer RLS, service principals, sized capacity. Scalable — but maintenance and capacity tuning are a standing engineering load.
L5	**Fully automated delivery platform.** Onboarding, isolation, scaling, monitoring and security are automated. New customers are configuration, not a project. **This is where DataTako operates.**

HOW TO USE THIS

Most teams sharing externally are at L1–L3 and feel the friction without naming it. The goal isn't to rush to L5 overnight — it's to stop solving the same problems by hand at every new customer.

13 · THE CHECKLIST

The scalable-delivery checklist.

Before you share another report externally, run it against this. If you can't tick a box, that's where your scale problem is hiding.

- **Model is decided, not defaulted.** You've consciously chosen App Owns Data over guest invites or publish-to-web.
- **End users need no Power BI license.** Authentication runs through your app via a service principal.
- **Row-level security is enforced per customer.** Customer A can never see Customer B's data, by design.
- **Capacity is sized to load, not guessed.** You picked an A/F-SKU based on expected concurrent usage.
- **You can pause or scale capacity.** Idle hours don't cost full price; spikes don't throttle.
- **Tenant and customer identities are separated.** External users never enter your Entra tenant.
- **Embed tokens are short-lived and server-generated.** No token logic or secret exposed to the client.
- **Throughput is monitored.** You watch capacity metrics and alert before users feel slowness.
- **Delivery is automated.** No human exports a PDF to onboard a new customer.
- **You priced year two.** Maintenance and Power BI/Fabric changes are in the budget, not a surprise.

> **FROM DATATAKO**
>
> Every unticked box is a future incident or invoice. Ticking all ten is the difference between "we embedded a report" and "we built a delivery platform that scales."

From chaos to scalable Power BI delivery.

DataTako helps software vendors, consultancies and data teams deliver Power BI to external users — scalable, structured, and fully under control. No per-user licenses for your customers. No throttling surprises. No manual exports.

If this checklist exposed a gap, that's exactly the conversation we have every day.

Talk to DataTako →

Book a demo
Scan to book —
datatako.com/demo

DataTako
datatako.com · paco@datatako.com · datatako.com/cost-calculator