PRODUCT-LED GROWTH

FOR ENGINEERS:

BUILD THE FUNNEL INTO THE PRODUCT

WHAT PLG ACTUALLY MEANS (AND WHAT IT DOESN'T)

At its core, a product-led growth strategy puts the **product itself** as the primary driver of customer acquisition, activation, retention, and expansion. Instead of a sales rep qualifying leads or a marketer nurturing cold emaiil sequences, **the product does the persuading**. Users sign up, discover value on their own, and convert when the product has earned it.

That sounds simple. **It isn't.** Many PLG implementations fail — and the pattern suggests that execution difficulty requires **fundamental company restructuring**, not just adding free trials.

Here's the part most PLG articles skip: the reasons most PLG implementations fail are **engineering failures.** Broken onboarding flows. Instrumentation bolted on after launch. Feature gates that behave inconsistently. Upgrade prompts that fire at the wrong moment. These aren't product management problems — they're **architectural decisions** made (or not made) at the code level.

If you're building a product from scratch, or rebuilding one with growth in mind, you have a genuine advantage. The patterns are well understood. What's hard is **executing them with the same discipline** you'd apply to a database schema or an API contract.

PLG

ONBOARDING FLOW ARCHITECTURE

SIGN UP
VERIFY EMAIL
WELCOME TOUR
CREATE WORKSPACE
ADMIN
MEMBER
IMPORT DATA
FEATURE GATE
CORE VALUE MOMENT
INVITE TEAM
USAGE EVENTS
UPGRADE PROMPT
NO
PLAN UPGRADE?
YES
BILLING SETUP
CONTINUED USAGE
PAYMENT SUCCESS

supramono
FOUNDER

SHIP VALUE

THE FUNNEL IS THE PRODUCT NOW

Traditional SaaS growth looked like this:
marketing generates leads, sales qualiifies them,
the product is the thing you get after you buy.
The funnel sat outside the product.

The product serves as your primary growth engine when acquisition and retention mechanics embed directly into core functionality.

ENGINEERS ARE THE PLG ARCHITECTS

WE BUILD **GROWTH MECHANICS** INTO THE ARCHITECTURE. TOOLS CAN'T ADD WHAT WASN'T DESIGNED.

ARCHITECTURE FIRST. GROWTH BUILT IN.

INSTRUMENT ONBOARDING FOR THE **AHA MOMENT.**

USAGE-BASED TRIGGERS

When a user completes a workflow for the fifth time, your event bus can trigger an in-app prompt at that exact moment.

Real-time, zero-latency growth signals.

IN-APP UPGRADE PROMPTS

Prompts fire at conversion moments — storage limits, team invites, premium exports.

The message, timing, and upgrade path are engineering decisions.

FRICTIONLESS ONBOARDING

Deliver value quickly. Remove friction. Guide users to their "aha moment" as fast as possible.

Treat first-run like any other critical path.

WHY TTV DRIVES EVERYTHING

aha!

TIME-TO-VALUE MEASURES HOW QUICKLY NEW USERS HIT THEIR "AHA MOMENT" — THE POINT WHERE CORE VALUE CLICKS.

TICK! TICK! TICK!

HIGH-PERFORMING SAAS TEAMS HAVE REDUCED TTV TO *HOURS, NOT DAYS.*

EVERY SECOND SHAVED COMPOUNDS INTO BETTER *ACTIVATION, RETENTION,* AND *FREE-TO-PAID CONVERSION.*

USER JOURNEY

KILL THE EMPTY STATE

Don’t show a blank canvas on first login. Seed the product with sample data, a demo project, or a pre-filled template so users experience core value **before** doing real work.

Notion does this. **Linear** does this.

The empty canvas is the **enemy** of time-to-value.

SURFACE FEATURES BY READINESS

Features should reveal themselves based on user readiness. Show one or two capabilities in onboarding. Gate the rest behind actions that signal intent.

Hide step three until steps one and two are complete — completion rates climb when the UI respects cognitive load.

PERFORMANCE IS A FUNNEL PROBLEM

A two-second delay in email verification is a funnel leak.

If analytics show 40% of users abandon signup due to slow load times, invest in performance optimisation.

This is infrastructure work with direct revenue impact.

TTV is undertracked — if you're not measuring it, you're flying blind.

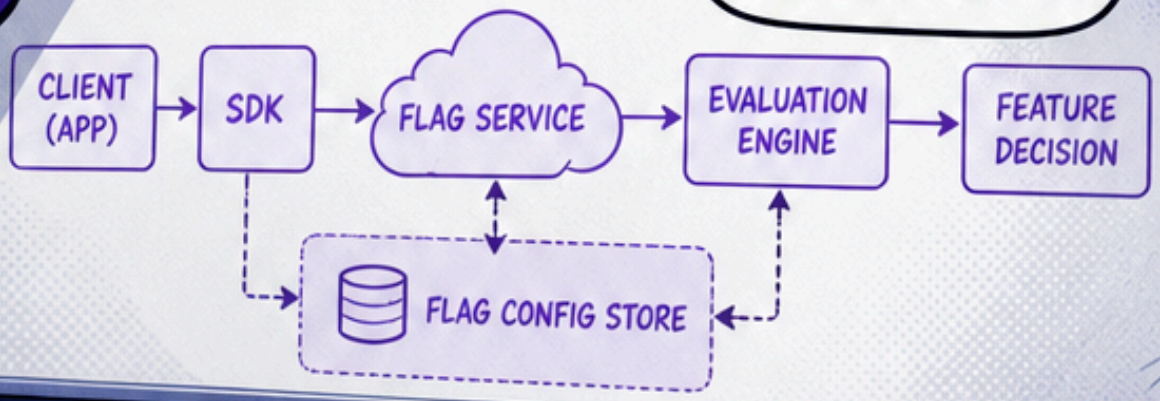
FREE TIERS ARE ENGINEERING DECISIONS, NOT PRICING DECISIONS

FREE TIER	PAID TIER
Basic Projects	Unlimited Projects
Limited Storage (5GB)	Unlimited Storage
Community Support	Priority Support
Basic Integrations	Advanced Integrations
Advanced Analytics (Locked)	Advanced Analytics
	Custom Domains
	Team Collaboration

ARCHITECT IT LIKE A **SYSTEM**, NOT A BUNCH OF IF CHECKS.

FREE TIER = AN **ENGINEERING** SYSTEM WITH THREE MOVING PARTS.

FEATURE FLAG ARCHITECTURE



1 FEATURE GATING

Decide upfront what sits behind the free/paid boundary. Enough value to see the value. Not enough to never need to.

Engineer it with **proper feature flags**, not scattered **if** checks.

2 TELEMETRY AT THE GATE

Every gate hit is a **conversion signal.** Capture who, what, why, and context. Personalise prompts. Track conversion by gate type.

3 UPGRADE PATH MECHANICS

The modal is a micro-sales page. Make the upgrade **one click, instant, and frictionless.** Engineering determines conversion rates, not just copy.

TAKEAWAY

ENGINEERS OWN THE USER JOURNEY, FROM FIRST VALUE TO PAID CONVERSION. **DESIGN THE SYSTEM, REMOVE FRICTION.** BUILD WHAT USERS ACTUALLY WANT TO UPGRADE TO.

VIRAL LOOPS DON'T HAPPEN BY ACCIDENT

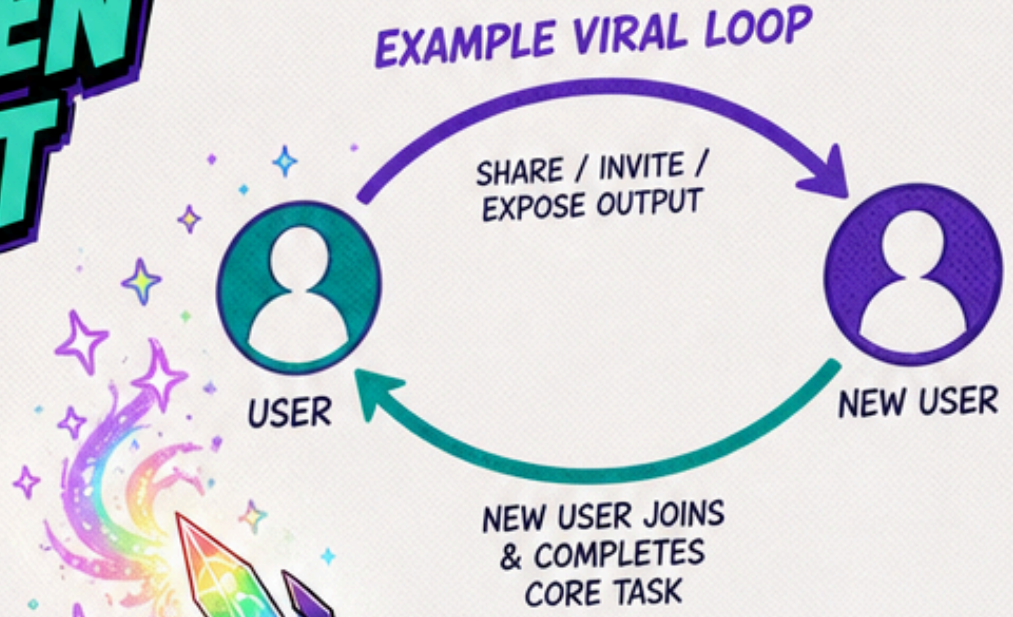
NETWORK EFFECT
The products that grow without marketing budgets — Slack in its early years, Figma, Calendly, Dropbox — all have one thing in common: using the product involves other people.

COMMUNITY: THE SOFT VERSION
Online communities drive advocacy, sharing, and referrals — powerful but not always scalable.

VIRAL LOOPS: THE HARD VERSION
Build the loop into the core workflow. When users complete the core task, the output should expose a new person to the product.

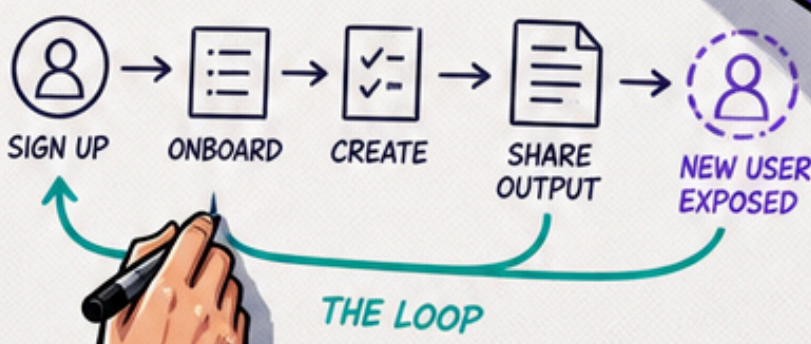
MEASURE EVERYTHING
Instrument, track, and A/B test every loop. Know the conversion rate from shared artifact to new signup.

OWN THE ENTIRE JOURNEY
From acquisition to expansion — compounding retention and LTV when done well.



ASK YOURSELF:

- ☑ What happens when a user finishes the core task?
- ☑ Can that output be shared, exported, or linked?
- ☑ Does it expose a new person to the product? Can the recipient clearly see what the product is?

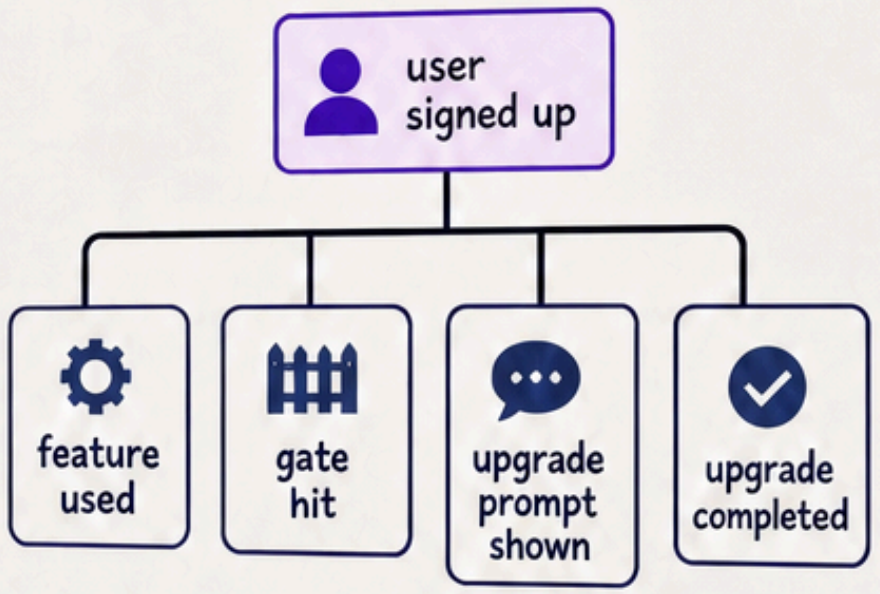


> "VIRAL LOOPS AREN'T LUCK. THEY'RE **DESIGNED**. BUILD THEM. MEASURE THEM. COMPOUND THEM.

EVENT TAXONOMY

BEFORE YOU WRITE A LINE OF PRODUCT CODE

Define your event schema — user signed up, feature used, gate hit, upgrade prompt shown, upgrade completed — before building. Naming conventions, properties, context objects. Treat it like an API contract, the way you'd write a database migration before application code.



EVENT PROPERTIES

- event_id (string)
- user_id (string)
- timestamp (ISO 8601)
- name (string)
- properties (object)
- context (object)

PROPERTIES (EXAMPLES)

- plan_tier (string)
- feature_name (string)
- source (string)
- value (number)
- currency (string)

CONTEXT OBJECTS

- user (object)
- account (object)
- device (object)
- session (object)
- geo (object)

Treat it like an API contract, the way you'd write a database migration before application code.

FUNNEL INSTRUMENTATION AS A DEPLOYMENT GATE

Every new onboarding step must have a corresponding analytics event before it ships. **No event, no deploy.**

This is a cultural norm as much as a technical one — and **engineers are the ones** who enforce it.

Wire retention cohorts to your pipeline: a drop in week-one retention should surface **before the sprint ends.**

PRODUCT QUALIFIED LEADS LIVE IN YOUR DATA PIPELINE

PQLs represent users who have demonstrated purchase intent through active product use.

*The PQL definition — which events, what frequency, what recency — **must live in your data pipeline** and feed your upgrade-trigger logic.*

*PQL-based outreach converts better than cold or MQL outreach because contact happens at a **moment of real intent.***

PQL EVENT CLUSTER DETECTED!

DEFINE IT. PIPELINE IT. TRIGGER IT.

DATA PIPELINE

PRODUCT EVENTS
- sign_up
- feature_used
- report_viewed
- settings_saved
- invite_sent
- ...

EVENT PROCESSING

PQL DEFINITION
EVENT: feature_used
FREQUENCY: ≥ 3
RECENCY: ≤ 7 days
PQL QUALIFIED

UPGRADE-TRIGGER LOGIC

OUTREACH ENGINE

PQL USERS

supramono Founder

10 YEARS STARTUP EXPERIENCE

YOU CAN'T SOLVE A PROBLEM YOU CAN'T SEE

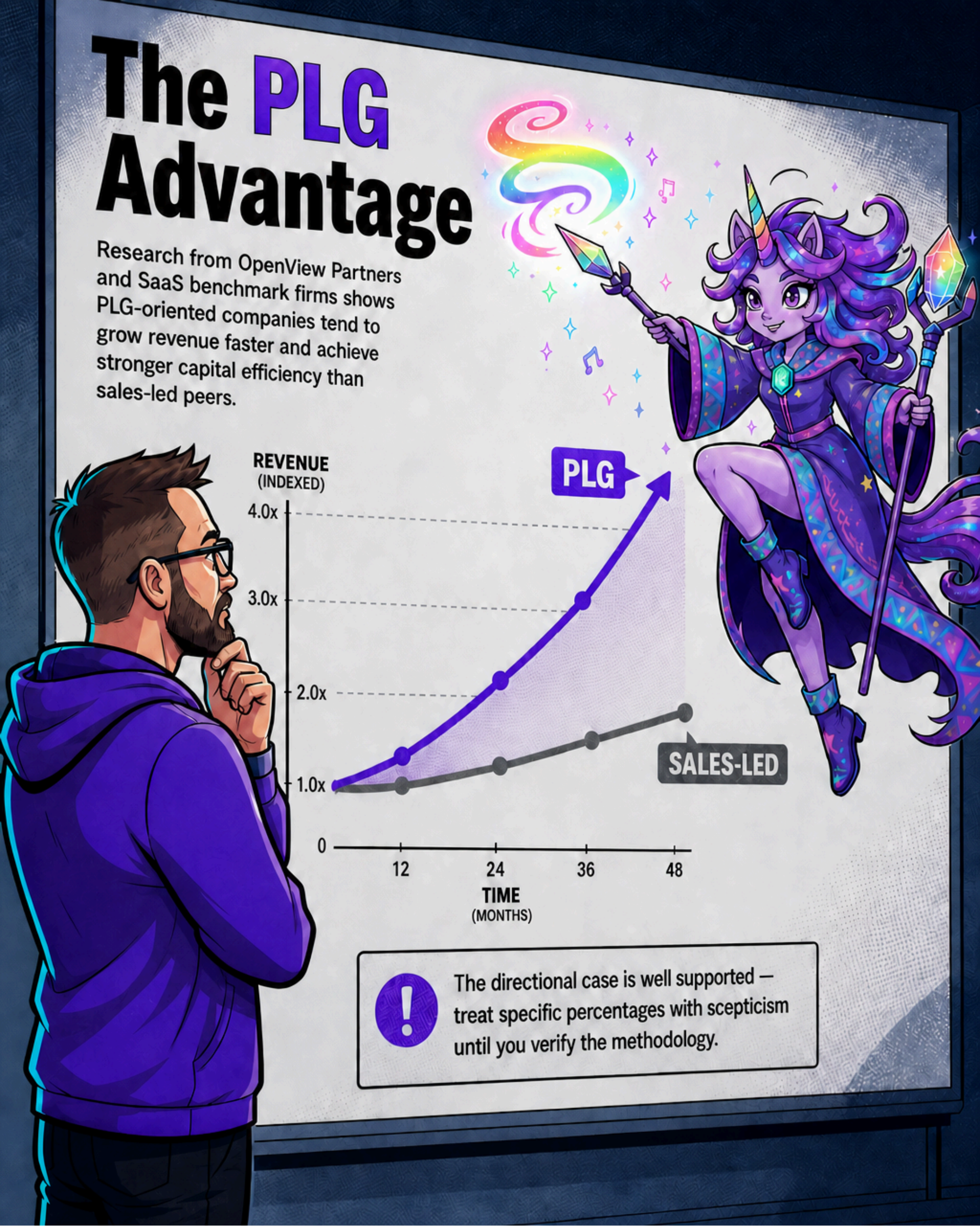
Most engineers ship first, then reach for analytics six months later — by which point there's no historical data and events are inconsistently named.

A significant share of users churn within **90 days.**

If observability isn't in place **before real users hit that window,** you're debugging in the dark with no baseline to compare against.

The PLG Advantage

Research from OpenView Partners and SaaS benchmark firms shows PLG-oriented companies tend to grow revenue faster and achieve stronger capital efficiency than sales-led peers.



The directional case is well supported — treat specific percentages with scepticism until you verify the methodology.

WHAT PLG ACTUALLY REQUIRES

The growth system lives inside the product: signup flow, onboarding, feature gates, upgrade triggers, viral surfaces, analytics pipeline. **All of it is engineering.**

If you're building solo or with a small team, your job isn't just to build the product — it's to build the growth mechanics too.

THE ASYMMETRIC EDGE

You don't need a growth team.
You don't need a marketing budget.
You need to treat growth mechanics with the same architectural rigour you'd give any other system — and instrument everything from day one.

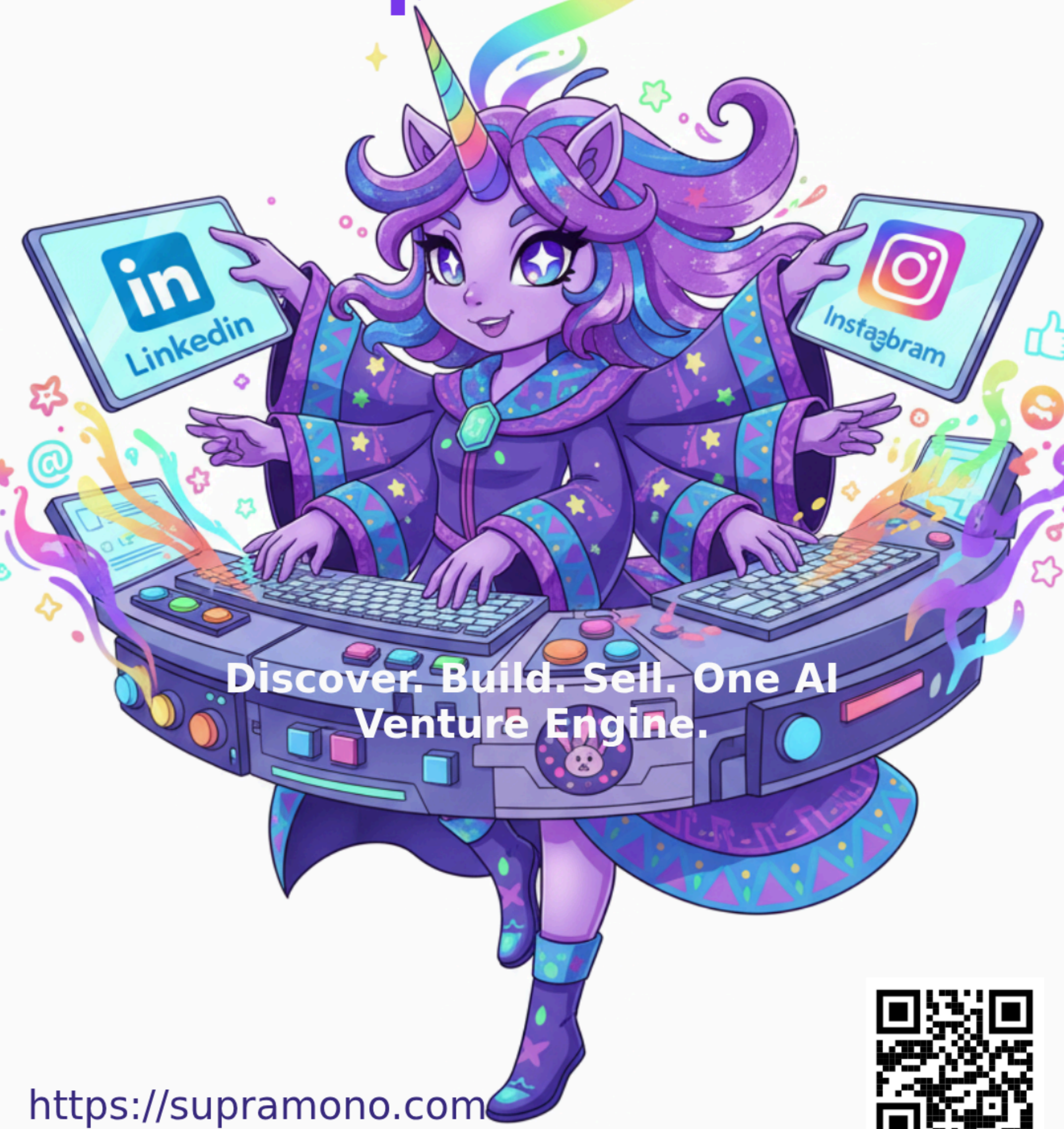
This is territory where technical founders win.

SOLO OPERATOR LEVERAGE

Building right now and thinking about growth without a team? Supramono is an AI venture platform covering opportunity discovery, product development, and go-to-market execution from a single interface.

See how a solo operator can do what used to take ten people.

supramono



Discover. Build. Sell. One AI Venture Engine.

https://supramono.com