Per Aspera Industries
Government Use
Portable Document Format
*UNCLASSIFIED

Per Aspera

Per Aspera — Antimemo

HOW TO BUILD YOUR 1ST FORWARD DEPLOYED ENGINEERING TEAM

Contributors

Mark Scianna (General Partner of Forward Deployed Venture Capital)
Dan Goldin (Editor in Chief at Per Aspera)

CONTENTS

INTRODUCTION

FOREWORD

Mark, the co-author of this Antimemo, is a former forward deployed engineer at Palantir who now runs Forward Deployed Venture Capital, fresh off of raising a $45M Fund II. Mark has seen "forward deployed engineering" from both sides of the table: first as one of its early practitioners, then as an investor helping his portcos create their own FDE strategies. Now, just like the rest of us in tech, he's seen the term permeate popular consciousness — often, deployed as a buzzword stripped of its original meaning and intensity.

Given that Mark now spends much of his time clarifying and reclaiming what Forward Deployed Engineering truly means, and that he built a fund that quite literally carries the name, we at Per Aspera couldn't think of anyone better to help bring this back to first principles – and clarify what Forward Deployed Engineering is, how it really works in practice, and whether the strategy makes sense for your company.

Let's rewind 15 years and start in Kandahar, Afghanistan.

Foreword by Mark Scianna

In the beginning, it was just us: two engineers dropped into a military base near Kandahar, handed minimal-but-clear marching orders from Palo Alto: 'Go there and win. Call if you need anything.' I spent months in Afghanistan, racking and stacking servers on classified networks, building products, and taking midnight helicopter rides to forward operating bases near the Pakistani border.

One night in the summer of 2011, while embedded with a Navy SEAL team, I read a blog post that a colleague sent over. The piece, Why Palantir Makes My Head Hurt, dismissed FDE as nothing more than "consultants with fancier titles." I had spent the week dodging rocket fire, building data integrations, training users, and debugging pesky databases. The take felt surreal. Just two months earlier, the U.S. successfully carried out the Bin Laden raid — the exact sort of mission Palantir was built to support.

Setting up our own satellite internet for comms back to HQ.

In the years since my deployments, Palantir practiced and perfected the art of putting engineers as close to the problem as possible. It's been nearly two decades since Shyam Sankar of Palantir gave this practice a name: Forward Deployed Engineering (or FDE, for short). Boy, oh boy, has that name stuck.

Today, it seems the tech world has been one-shotted by FDE's seductive promise. Plenty are talking the talk, but who is doing the real work? Anyone selling you an "FDE overnight" scheme is a false prophet. The work is hard, the stakes are high, and it's not always a good fit.

But, if this is the game you intend to play, this playbook will show you how to do it right.

Mark with his colleague Matt Hampton en route to FOB Ripley in Tarin Kowt, via Kabul.

SECTION 001

DESIGNING AGAINST FUNDAMENTAL PHYSICS

FDE has become the hottest job in tech, and it's happened so quickly that the role faces a recruiting bottleneck. Y Combinator's job board has 100+ FDE roles listed, up from zero a couple years ago. Agentic AI companies are adopting the model, and OpenAI will send you an FDE once your bill crosses $10M/year. Microsoft's Satya Nadella has lavished praise on the concept, as has Mark Benioff, whose Salesforce recently started posting its own "forward deployed" JDs. The Fortune 500 and too-many-to-count startups are piling in, slapping the titles on teams without truly empowering them.

No wonder the term has been thoroughly meme-ified, both in the internet sense but also in the truer, more fundamental mimetic one: FDE is an idea spreading without its meaning fully intact.

(For Mark, it actually was Iraq, and Afghanistan...)

Ironically, the earliest critiques of Forward Deployed Engineering have become a self-fulfilling prophecy and taken on some truth. But FDE, as pioneered and practiced by Palantir, is no consultancy wrapper nor sales in a trenchcoat. For the Palantirians dispatched from the artist's colony to the proverbial (or real) front lines, it is a rigorous discipline. Engineers embed deeply within a customer's organization, wield unusual autonomy, and bear full accountability for driving tangible, operational outcomes.

Watch The FDE Playbook for AI startups with Bob McGrew Video: https://youtu.be/Zyw-YA0k3xo

---

McGrew defines a forward deployed engineer as someone "who sits at the customer site and fills the gap between what the product does and what the customer needs." We'll use that as our starting point, and "Sparkling Sales Engineering" as our term for everything else that is...not FDE.

Our first order of business is to distinguish what's Forward Deployed vs. Sparkling Sales Engineering.

SECTION 002

PUT ENGINEERS AS CLOSE AS POSSIBLE TO THE PROBLEMS THEY'RE SOLVING

This first point isn't novel. But we must start here because it's the one that teams most often forget (or subordinate to outside forces).

From the outset, we kept the ISS operating plan anchored to the physics of low Earth orbit (LEO). Physics held veto power over the plan before anything else. That sounds simple enough, but you'd be surprised how often budgets, politicians, unfounded ambition, or schedule pressures try to renegotiate with thermodynamics.

History is clear: building to the physics is not a given. It's a management choice you must enforce every day. Never subordinate physics to org charts or other outside pressures.

Palantir's ruggedized server kit in Tarin Kowt, Afghanistan

"If I could sum FDE up in one sentence," Mark says, "it's that Palantir really believes you should put an engineer as close to the problem as possible."

Below, we'll offer an acid test that will help you tell FDE apart from Sparking Sales Engineering.

001 // Are you sending actual engineers?

FDE means real software engineers embedded with the customer. Not PMs, technical consultants, or salespeople disguised as engineers. Someone who opens an IDE daily, builds custom integrations on site, and delivers on the order of days to weeks. Rule #1: If your "FDE" hasn't built something in the last couple of weeks, it's not FDE.

002 // Is the commitment sustained?

You need deep immersion, frequent discomfort, and a sustained presence. Quick site visits don't count. Rule #2: If you're just parachuting in for a Mon-Thu trip, it's unlikely to be FDE.

> "My first Kandahar deployment lasted three months because we had so much to learn. Even as the product matured and our expertise grew, meaningful deployments took weeks, not days. You need time to understand the customer's workflow, build trust with users, and iterate on solutions based on real feedback."
>
> Mark Scianna — General Partner of Forward Deployed Venture Capital

003 // Is the engineer empowered to work on the fly?

This may be the truest litmus test. FDE runs on the military concept of Auftragstaktik, where HQ sets the objective and delegates tactical execution almost entirely to the field team. Recall Mark's marching orders: Go there and win. What came next was up to him: which data to migrate, who to train, how to expand within the organization, and so on. Day-to-day decisions belong to the FDEs, who phone home only when critical technical backup is needed or a true blocker arises.

Rule #3: autonomy is non-negotiable.

Mark (not pictured) flying with his colleague Tri Tang in the back of a C-130.

004 // Is the field feeding the product?

FDE is a product discovery loop, not an implementation service. The field surfaces real workflows, ships gravel-road solutions that prove value, and hands patterns to HQ that can be abstracted, hardened, and paved into the platform. Done right, field work raises your product leverage – done lazily, you'll just pile up bespoke work.

Early on, the ratio will tilt towards more bespoke field work. Your FDEs build more stuff from scratch. The feedback loop will feel non-linear and lumpy. But...if you successfully capture and abstract the patterns, marginal deployments get easier, cheaper, and more valuable.

Here are the mechanics of how this should work (c/o Bob McGrew):

- Start with one outcome. Pick a top-five business outcome for your customer and build backwards from it. You're shipping results, not "an install."

- Demo as spec: Work quickly (≤10 days) towards a live demo that relies on real data and creates desire for your user(s). This focuses the build, exposes the seams you must close next, and aligns everyone on "this is the thing."

- The two hats. Echo and Delta are the yin and yang of Palantir's model. Echo teams are domain insiders with a rebellious streak, embedded with customers to unearth real, high-leverage problems and manage relationships; Delta teams are rapid-prototyping engineers who build and deploy fast, working hand-in-hand with Echo to ship the scrappy v0 in weeks, not quarters. In practice, these folks were/are unicorns that usually have customer experience and an engineering background.

- Build the gravel road: Discover and construct the minimum viable path to the outcome that moves the needle/solves the user's pain point. Fast, a bit ugly, but true — and this will expose patterns (reusable primitives, inputs/outputs, workflows, feature sets, knobs) you may later pave into the platform.

- Know your second user. Your fellow FDEs are your second customer, the first being the end user. FDE #1 leaves something FDE #2 can build off of. #3 refines it further, and by FDE #N, it's the default path. If field engineers will not choose your abstraction over a one-off, it is the wrong abstraction.

- Abstract one level up. Product's job is to generalize what worked from a deployment into something more general than the site request.

- Prove reuse. First across a second logo, then across a second segment, then, as a generalized product offering.

- Promote to platform: Pull proven concepts into your core products so the next deployment is more configuration, less net-new code.

Most field-born discoveries — 90%, perhaps — won't survive beyond first contact (e.g, one customer site or user persona). But those that break through are hard-won revelations crystalized from the crucible of friction. These bits of tribal knowledge, user insights, behavior patterns, features, and requirements are fed back to the mothership, where they are swiftly captured, meticulously distilled, and ultimately woven into the technology that powers Palantir's internal and external tools.

If it's working, in a healthy loop, revenue per FDE will rise as software replaces the engineers who are bridging the gap between what the product can do and what the customer needs. When it's not working, your economics tell on you.

SIGNS IT IS WORKING: Time–to–first outcome falls by segment. Field effort per outcome falls across customers. A rising share of field code in core. Redeploys rely on configuration, not new code. Breakout new products surface from on-site discovery.	**SMELLS OF A SERVICES SHOP:** Bespoke solutions are artisanally built and rebuilt at multiple sites. Long-lived forks sit in customer branches. Implementation backlog dwarfs platform backlog. No flags, no telemetry, no kill dates.

Alas, we finally have Rule #4: If Signals aren't trending right and Smells persist — revenue per FDE flatlines, and custom work continues accumulating — you may be running a services shop.

Do Things That Don't Scale, At Scale

Palantir's true strength, then, wasn't strictly forward deploying engineers. It was capturing rare, resonant customer insights and abstracting them into their core technology stack. Palantir cracked the arithmetic of "doing things that don't scale, at scale," as McGrew puts it.

FDE "builds the product customers need, when they need it, at the edge," per Palantir Forward Deployed Architect Chad Wahlquist. "The feedback loop from FDEs experiencing customer pain at the edge and turning it into product allows us to cycle and build the right product much faster."

Most miss this, and do not discover the patterns, frameworks, and trade spaces that define what should exist.

And remember: this is painful. In McGrew's recent appearance on the YC pod, we counted eight mentions of "pain." If you master the loop, you can achieve power-law outcomes. But it's painful and hard to pull off. You can easily veer into customer support or consulting, which dilutes the strategy and balloons your cost structure.

Which brings us to our next point: *how much does FDE cost?*

SECTION 003

THE UNFORGIVING ECONOMICS

FDE changes who you hire, your cost structure, how you organize teams, your development cadence, and product development. It's a high-variance strategy: high effort, high risk, high potential reward.

Colin Anderson, Palantir's former CFO, is blunt about the math. Companies attempting FDE on small contracts are "lighting equity on fire," he said recently on TBPN. "A world-class engineer plus their equity dilution is an expensive check...You have to be a capital allocator to pursue this model."

FDE requires a rare breed.

Elite, high-agency engineers who can code, read the room, and to some extent, sell. On engineering parity alone, these folks could easily work at Google or Meta. But they are not ICs or SEs with nicer badges; they must wear many hats. They need product intuition, creativity, and customer-facing savvy. This is a hard, expensive full-stack skillset to hire for. So, you should expect longer hiring cycles and higher comp.

Know thy burn & unit economics.

Seven-figure ACV contracts can generally support the FDE model; eight or nine figures obviously can. But smaller contracts, say in the $20,000 to $100,000 range, don't pencil unless explicitly viewed as R&D toward a bigger contract. Force-fitting FDE into modest deals = cash burn or dilution to subsidize delivery.

The natural filters for FDE

1. **Large Enterprise Deals.** You're hunting for ≥$1M deals, or a truly strategic pilot with a clear line of sight to that scale.

2. **Significant Learning Requirements.** You're facing unknowns you can't discover remotely. Some problems or learnings are physically constrained, so you must go there and learn. If workflows, data realities, and constraints are opaque, and quick site trips or stop-and-go dialogue will not surface the truths you need, you may have the sufficient conditions.
3. **Complex, Mission-Critical Problems.** Defense, aerospace, critical infrastructure, energy, heavy industrial automation...what's the throughline? They are all places where reliability, precision, and performance are table stakes, and they are typically all great fits for the FDE model. Customization isn't optional, but required, and conservatism rules the roost. You need someone who can help bridge old and new ways.

Full-on FDE makes sense when three conditions hold: you sell into markets that can bear very high prices, you admit you don't fully know the product and will discover it bottoms-up with users, and your platform is built for extensibility with discipline about what stays at the edge versus migrates to core.

An example of one of Palantir's commercial accounts: Airbus + Palantir impact study

---

It's no coincidence Palantir concentrated on USG, national agencies, aerospace and defense, and the Fortune 500. This customer set has a high willingness to spend and can carry the weight of forward deployment.

SECTION 004

WHY MOST COMPANIES CAN'T STOMACH THIS

The Tactical Operation Center / customer's HQ where Mark set up Palantir systems in 2010.

Unit economics aside, FDE demands a rare alignment — the perfect alchemy of complexity, scale, and culture.

FDE inverts the org chart. Instead of HQ roadmaps and PM committees, authority and trust is pushed to the edges. Palantir's internal doctrine became "radical deference to teams in the field."

> Few companies are willing to fully subordinate their roadmap to field teams...this is an anathema to most cultures, where leadership creates top-down roadmaps and plans.
>
> Barry McCardel — Cofounder/CEO of Hex and ex-Palantir Deployment Strategist

The gag reflex: Most organizations simply can't stomach this model. The average corporation, organization, or bureaucracy wants routine check-ins, approvals for every technical choice, predictable roadmaps, and tight budget controls. FDE affords no such creature comforts.

Field teams might make calls that HQ would never make. They'll be wrong, early and often. And that's the point!

The gut check: If you can't tolerate ambiguity or code that starts ugly and gets good, or every v0 needs PM sign-off, or failure is punished instead of studied, don't do FDE.

You can't get FDE's compounding upside without stomaching the initial pain and disorder. If you want certainty, you can run a more traditional services playbook. If you want discovery, push power to the edge and learn to manage the mess.

SECTION 005

GO FORTH, FORWARD DEPLOYED ENGINEER

So, you've got the context — maybe more than you bargained for.

Now for the question you're all wondering: Is FDE right for my company?

The two-word answer is, of course, maddeningly frustrating. It depends. On your business model, your time horizon, your market, and your appetite for pain. FDE can be a moat-builder or a margin-killer. It can seed a deca-billion-dollar business or drain your balance sheet.

Ultimately, this is what you should be considering when deciding whether FDE is right for you.

1. **Upfront Cost vs. Later Payoff:** The rule of thumb is to not forward deploy for <$1M ACV deals (unless it is an investment, which credibly leads to $1M+ outcome). The teams that make FDE work (Palantir, Anduril, etc.) were well-capitalized and/or selling largely from the incorporation womb. If your board demands SaaS-like margins in 24 months, the invest now, harvest later approach may be difficult to justify. You, the founder, must judge: can we invest heavily to unlock a possibly huge market 5+ years out? If yes, then FDE may be right for you as a go big/go home bet with power-law upside. (Case in point: $PLTR).

2. **Retention and Talent Development:** You need to be a hell of a recruiter. While you'll be hard-pressed to get any good public data on this, the anecdata is clear: attrition must be managed. It's a doubled-edge sword: the field attracts ambitious builders who want to make an immediate impact. They run hot. Even on the right mission, churn is real, whether from burnout, spinning out to found a startup, or moving into leadership. Design for that reality. FDE mints founders, so treat it as a leadership pipeline, not a forever seat. Run defined tours, keep a bench, and write the succession plan before you need it. If you cannot recruit, develop, and backfill at a brisk cadence, you can't sustain FDE.

3. **Headcount Ratios:** Plan for a forward-heavy mix. Palantir carried a disproportionately large share of engineers in forward roles relative to pure product roles (which is unusual vis-a-vis most software companies). If FDE is core to your model, expect something similar. You may want to plan for roughly a 1:1–1:2 ratio of forward engineers to core engineers. The optimal ratio depends on situational factors, like stage. Early is lopsided and may look like 1:0 for a stretch (all hands on deck with customers). As the product hardens, you might leverage a smaller, more agile FDE team feeding a larger platform group. A forward function lives in COGS (even if it's actually R&D) and compresses gross margin while discovery is heavy. If you cannot budget for a persistently higher services headcount, it will be difficult to employ the FDE model.

4. **Investor Relations:** VCs and institutional investors like traditional SaaS in part because it optimizes for a very low headcount to keep margins high. FDE calls for you to fund a larger, in-house, services-like team because you treat it as R&D. This will show up in the P&L: gross margins compress while deployment is heavy, then trend up only as field patterns are productized. Set and manage expectations early on, with yourself, prospective investors, future controllers, your board, and the market. State plainly that the forward-heavy mix is a deliberate R&D investment, and show how you're trading near-term margin drag for a differentiated product development strategy. Practice your story, practice it some more, and perfect it. If you need to spell it out for stakeholders, start with this and turn it into your own:

We're investing more in forward deployed engineering now (by design), which will depress margins in the near term, so we can convert field wins into reusable product development insights that accelerate deployments, lift expansion, and grow margins over time.

Make the "why" explicit, show the metrics, and, if it helps, send that weary investor this piece.

5. **Define, Live, and Die by your KPIs:** The metric set to watch: conversion, expansion, and time-to-value. FDE, done right, helps you land bigger deals and expand them faster (the logic being that by delivering value quickly, you grow a pilot to a $1M+ deal in a way pure sales could not). Palantir often started small with projects that turned into huge contracts (e.g., Project Maven) after value was proven on the ground. The beauty is that your FDEs are your key lever for account expansion. Once embedded, the engineer finds more problems to solve and essentially creates their own upsell opportunity by delivering results. If you're seriously considering going down this road, have a good handle on pilot conversion rate, expansion revenue, and time to value for customers under an FDE model vs. normal. With a successful FDE strategy, you'd expect shorter time to first value and high expansion multiples on initial deals.

6. **You have to be customer-obsessed, and your customer has to be obsessed:** The point of FDE is stickier, higher-value outcomes. That is its raison d'être. While harder to quantify, things like customer retention/churn or NPS score can reflect this. Palantir's retention on core accounts and strategic programs speaks to this point. And quantitatively, Palantir's improving margins showed progress: as the software matured, the company could reduce how many engineers were sitting with a given client. But, a word of caution...if a customer's environment changes or if you step back, will they maintain success? Over a 5–10 year span, a well-executed FDE model should see decreasing marginal cost per customer (as more of the solution is out-of-the-box) while maintaining/improving outcomes. If instead you still need just as many custom engineers for the Nth customer as you did for the first, the model isn't scaling.

The Tactical Operation Center / customer's HQ where Mark set up Palantir systems in 2010.

Conclusion

FORWARD, TO THE FRONTIER

What does all of this mean for the world of hardware, which is Per Aspera's obsession and the work that many of you do every day?

The frontier has always belonged to people who sit closest to the problem. From field engineers who kept mainframes alive to astronauts strapping in unproven machines, progress came from those who chose immersion over abstraction. At its best, Forward Deployed Engineering is that same instinct: put talent, judgment, and will where the stakes are highest. This instinct is bigger than any one name, and it's how the hardest problems get solved.

FDE is a choice about how you build. Done well, discovery becomes a rhythm — land, expand, abstract — and the product starts to carry the weight. But it is not for most teams. It asks for capital, patience, and a high pain tolerance.

If you're in hardware, and you've been doing something that looks and feels like FDE, this Antimemo should arm you with sharper language, cleaner metrics, and a clearer operating model. If you are new to this game, hopefully this helps clarify whether this is the right rodeo for you. We've given you the right accounting so that you do not have to FAFO and learn the true costs of misapplying this model the hard way.

Decide which game you are playing, and then commit.

Mark in Tarin Kowt, Afghanistan

Two Calls to Action

1. If you are doing this type of work and you have the scar tissue and receipts to show for it, we want to hear from you. Send the Per Aspera team your lessons, near-misses, and quiet compounding wins. And sign up for Per Aspera, if you aren't already!

2. If you're a founder who wants an investor who can help you build a real forward-deployed function, reach out to Mark Scianna at Forward Deployed VC. He has sat in the seat and now backs teams who can stomach the pain. If you intend to play this game, bring people who will sit closest to the problem with you, and stay there until the damn thing works!

NOTHING FOLLOWS