— FIELD NOTES · DIGITAL WORKBOOK

SWE → FDE
THE 90-DAY TRANSITION ROADMAP.

ambiguity, ownership, and the work that happens beyond staging

How to use this roadmap.

This isn't a listicle of tips. It's a 12-week, checkbox-driven program that rebuilds how you work — not just what you know.

▸ WHO THIS IS FOR

Software engineers who want to move toward Forward Deployed Engineering — whether that's a Palantir-style enterprise/gov role, a traditional solutions/ delivery track, or one of the newer AI-native FDE roles at places like OpenAI, Anthropic, or fast-moving startups. The core discipline is the same across all three; only the surface-level tech stack changes.

▸ THE STRUCTURE

PHASE 1 · DAYS 1–30

Foundation. Rebuild the underlying skills: ambiguity tolerance, ownership, domain immersion.

PHASE 2 · DAYS 31–60

Simulate the role. Do the actual work — end to end, on a project you fully own.

PHASE 3 · DAYS 61–90

Position and land. Translate what you built into language hiring managers recognize, then go get interviews.

▸ HOW TO WORK THROUGH EACH WEEK

- [ ] Read the week's framing once, fully, before doing anything.
- [ ] Do the exercise for real — on a real codebase, a real project, not hypothetically.
- [ ] Fill in the prompt boxes by hand. The writing is where the thinking happens.
- [ ] Don't skip weeks because they feel “too basic.” The basics are the differentiator.

What FDE actually is (and where it came from).

Most explanations of this role start with AI. That's backwards — and it'll give you the wrong mental model.

The Forward Deployed Engineer role was created at Palantir in the early 2010s, internally named "Delta." Palantir was building software for intelligence agencies — environments where the problems were too sensitive and too specific to be handed off through a normal requirements-gathering process. Someone needed to sit inside the customer's world, understand the actual operational problem, and build production systems under the real constraints the customer faced. That someone became the Forward Deployed Engineer.

Palantir built this around two paired teams: **Echo** teams, who were domain experts — often from the customer's own industry (military, healthcare, finance) — whose job was to understand the real problem, and **Delta** teams, the engineers who built the system around that understanding. By 2016, Palantir actually had more FDEs than conventional software engineers.

> **The core discipline:** own a problem end-to-end — scoping, architecture, delivery, and what happens after it ships — inside an environment you don't control and can't fully predict. AI didn't invent this. It just made more companies need it, faster.

▸ THE THREE FLAVORS YOU'LL SEE TODAY

FLAVOR	ENVIRONMENT	WHAT'S DIFFERENT
Enterprise / Gov	Legacy systems, defense, regulated industries	Long engagements, heavy domain immersion, security clearances
Traditional Delivery	SaaS platforms, complex B2B implementations	Custom integration + configuration at scale
AI-Native	LLM/agent-based products (OpenAI, Anthropic, startups)	Faster cycles, evals and model behavior added to the toolkit

This roadmap builds the discipline that's common to all three. Phase 3's Appendix has an optional module if you're specifically targeting the AI-native track.

— PHASE 1 · DAYS 1-30

FOUNDATION.

rebuild how you work, not just what you know

Four weeks. One goal: get honest about the gap between how you currently work and how the role actually demands you work — then start closing it.

The honest skill audit.

Not a tech-stack checklist. An audit of ambiguity tolerance, generalist range, and ownership mentality — the traits the role actually selects for.

▸ SELF-ASSESSMENT

Rate yourself honestly (1–5) on each. Don't round up.

TRAIT	WHAT IT LOOKS LIKE IN PRACTICE	SCORE
Ambiguity tolerance	Can you start on a problem with no clear spec?	___ / 5
Ownership mentality	Do you follow a problem past “my part” is done?	___ / 5
Domain curiosity	Do you dig into how the customer's world actually works?	___ / 5
Generalist range	Can you move across an unfamiliar stack without panic?	___ / 5
Communication under pressure	Can you explain a tradeoff to a non-engineer, live?	___ / 5

▸ REFLECTION

WRITE IT OUT

Which score surprised you? What's one specific moment from your work in the last 6 months that explains that score?

WRITE IT OUT

If an FDE hiring manager shadowed your last sprint, what would they see that concerns them?

Ambiguity navigation drill.

This is the muscle the Echo/Delta model was built around: taking an underspecified problem and scoping it yourself, before anyone hands you a ticket.

▸ THE EXERCISE

Find a real, messy problem this week — at work, in an open-source project, or in your own side project. It should have **no clear ticket and no clean requirements.** Examples: a vague bug report with no repro steps, a customer complaint with no technical detail, a "can you look into why this feels slow" Slack message.

- [ ] Resist the urge to ask for more detail immediately. Sit with the ambiguity for at least 30 minutes.
- [ ] Write down 3 different ways the problem could be interpreted.
- [ ] Pick the interpretation you think is most likely to matter to the person who raised it — and justify why in writing.
- [ ] Scope a plan of attack before writing any code.
- [ ] Only now, go verify your interpretation with the person who raised it.

▸ REFLECTION

WRITE IT OUT

How close was your first interpretation to what actually mattered? What signal did you miss or catch?

Domain immersion.

FDEs aren't hired for API knowledge. They're hired because they understand how the humans on the other side of the software actually work.

▸ THE EXERCISE

Pick one real industry or workflow you don't currently understand well — logistics, healthcare claims, defense operations, insurance underwriting, trading operations, whatever's accessible to you. Spend this week going deep on how the actual humans in that workflow do their jobs, not on the software they use.

- [ ] Find and read 2–3 first-person accounts from people who work in that domain (forums, interviews, Reddit, YouTube).
- [ ] Map out their workflow step by step, in plain language, no jargon.
- [ ] Identify 3 points in that workflow where the humans are clearly working around bad tooling.
- [ ] Write one paragraph explaining the domain to someone who's never heard of it.

WRITE IT OUT

What surprised you about how this domain actually works, versus how you assumed it worked from the outside?

Unfamiliar codebase reps + checkpoint.

Working in code you didn't write, in a system you don't fully understand, is the default condition of the job — not the exception.

▸ THE EXERCISE

- [ ] Pick an open-source repo you've never touched, ideally 10K+ lines.
- [ ] Give yourself 90 minutes to find and fix (or draft a fix for) one real open issue — no tutorials, no hand-holding.
- [ ] Time how long it took you to form a mental model of the codebase's structure.
- [ ] Submit the PR, or write it up as if you were going to.

▸ PHASE 1 CHECKPOINT

Before moving to Phase 2, answer honestly:

CHECKPOINT

Did this month feel like exercising a new muscle, or did it feel like things you already do well? If the latter, you may be closer to ready than you think — move faster through Phase 2. If the former, that's normal. Keep the pace.

PHASE 2 · DAYS 31-60

SIMULATE THE ROLE.

do the actual work, not a rehearsal of it

You can't interview your way into this role on theory. This phase builds one real, end-to-end artifact that proves you can own a problem the way an FDE does.

Reverse-engineer real job postings.

Job posts are the clearest signal of what companies actually mean by "FDE" — and the meaning shifts by track.

▸ THE EXERCISE

- [ ] Pull 8–10 real FDE / Forward Deployed Software Engineer postings — mix enterprise (Palantir-style), traditional delivery, and AI-native (OpenAI, Anthropic, fast-moving startups).
- [ ] For each, extract: required skills, described day-to-day, and what “success” sounds like in the posting.
- [ ] Group postings into the 3 tracks from the Context section. Note what's genuinely different, not just differently worded.
- [ ] Build a single skill map: which skills are common across all 3 tracks, which are track-specific.

WRITE IT OUT

Based on this research, which track fits your background and interests best — and why?

Ship one end-to-end mini project.

This is the centerpiece artifact of the whole roadmap. Everything in Phase 3 gets built on top of it.

▸ THE BRIEF

Pick a real, specific problem — ideally for an actual person or small business, not a toy dataset. Simulate a full embed: you scope it, you build it, you deploy it somewhere real (not just on your laptop), and you stay accountable if it breaks.

- [ ] Scope the problem yourself — talk to the actual person who has it, if possible.
- [ ] Build and ship a working solution. Deployed and usable beats clever and unfinished.
- [ ] No staging environment as a safety net — ship it the way an FDE would, into something real.
- [ ] Document your scoping decisions and tradeoffs as you go — you'll need this for Week 10.

> **This does not need to be technically impressive.** It needs to demonstrate ownership: you found the problem, you scoped it, you shipped it, and you can explain every decision you made along the way.

Mock incident and production-fire drills.

The job includes debugging live, in front of the person affected, while staying calm enough to explain what you're doing.

▸ THE EXERCISE

- [ ] Recruit a friend, mentor, or peer to play “customer.”
- [ ] Deliberately break something in the project you shipped in Week 7.
- [ ] Debug it live, on a call or screen-share, narrating your reasoning the whole time.
- [ ] Have your “customer” interrupt with questions and pressure — that's the point.
- [ ] Run this drill at least twice this week.

WRITE IT OUT

Where did you lose composure or clarity? What would you do differently next time you're debugging with someone watching?

PHASE 3 · DAYS 61-90

POSITION & LAND.

translate what you built into language hiring managers recognize

You've built the skill and the proof-of-work. This phase turns it into a resume, a story, an outreach plan, and interview reps.

Rewrite your resume into FDE language.

SWE resumes lead with tech stack. FDE resumes lead with ownership and ambiguity resolved.

▸ BEFORE / AFTER PATTERN

SWE FRAMING	FDE FRAMING
“Built a feature using React and Node.js”	“Owned [problem] end-to-end for [customer/team], from scoping through deployment”
“Fixed bugs in production”	“Debugged and resolved a live production issue directly with the affected team, under time pressure”
“Worked with cross-functional teams”	“Translated an ambiguous, undocumented business problem into a scoped technical solution”

▸ THE EXERCISE

- [ ] Rewrite your 3 strongest resume bullets using the pattern above.
- [ ] Add your Week 7 project as a headline bullet — problem, ownership, outcome.
- [ ] Rewrite your LinkedIn headline to lead with ownership/ambiguity, not tech stack.
- [ ] Get feedback from one person already working in an FDE-adjacent role, if you can find one.

Turn your project into a case-study narrative.

Interviewers remember stories with a clear arc. Give them one.

▸ THE NARRATIVE ARC

BEAT	WHAT TO COVER
1. The problem	What was actually broken or missing — in the customer's words, not yours
2. The ambiguity	What wasn't specified, and how you scoped it yourself
3. The build	What you shipped, and the one hardest tradeoff you made
4. The outcome	What changed for the person or system — concretely
5. What you'd do differently	Shows judgment, not just execution

▸ THE EXERCISE

- [ ] Write the 5-beat narrative in under 400 words.
- [ ] Cut it down to a 90-second spoken version. Time yourself saying it out loud.
- [ ] Practice it on 2 people who don't know the technical details — can they follow it?

Targeted outreach.

Enterprise/gov, traditional delivery, and AI-native tracks hire through different channels — outreach that works for one often flops for another.

▸ BY TRACK

TRACK	WHERE TO FOCUS OUTREACH
Enterprise / Gov	Direct company career pages, referrals, industry-specific networking (defense/gov contractor circles)
Traditional Delivery	Sales engineering / solutions / customer engineering teams at established B2B SaaS companies
AI-Native	Company blogs and careers pages (many describe the role directly), warm intros via AI-community networks

▸ THE EXERCISE

- [ ] Build a list of 15 target companies across your chosen track(s).
- [ ] Send 10 personalized outreach messages this week — reference your Week 7 project and Week 10 narrative directly.
- [ ] Ask 3 people already in FDE-adjacent roles for 15-minute conversations.

Interview prep.

FDE interviews test the same thing the whole roadmap has been building: can you scope ambiguity and own an outcome, live, under pressure.

▸ QUESTION TYPES TO DRILL

- [ ] “Here's a vague customer problem — how would you scope it?” (ambiguity navigation)
- [ ] “Walk me through a time you owned something end-to-end.” (your Week 10 narrative)
- [ ] Live debugging or system-design rounds in an unfamiliar codebase or system
- [ ] “How would you explain [technical tradeoff] to a non-technical stakeholder?”

▸ THE EXERCISE

- [ ] Run 3 mock interviews this week — at least one live-debugging style.
- [ ] Record yourself answering the ambiguity-scoping question. Watch it back.
- [ ] Prepare 2–3 sharp questions to ask them about how they structure the FDE/customer relationship — it signals you understand the role.

> You've built the proof-of-work, the narrative, and now the reps. This is the finish line of the roadmap — not of the work itself.

Resume bullet & LinkedIn templates.

Fill-in-the-blank starting points — adjust to your actual project and voice.

▸ RESUME BULLET TEMPLATE

TEMPLATE

Owned [specific problem] end-to-end for [customer/context] — from ambiguous initial ask through scoping, build, and production deployment, resulting in [concrete outcome].

TEMPLATE

Debugged and resolved [issue] directly with [affected team/customer] under live production conditions, restoring [system/metric] within [timeframe].

▸ LINKEDIN HEADLINE TEMPLATE

TEMPLATE

Software engineer who owns ambiguous problems end-to-end — from [domain/context] scoping through production deployment. Currently exploring Forward Deployed Engineering roles.

▸ OUTREACH MESSAGE TEMPLATE

TEMPLATE

Hi [name] — I've been building toward Forward Deployed Engineering and just shipped [one-line project description], where I [scoped/debugged/owned] [specific detail]. I'd love 15 minutes to learn how [company] structures the FDE/customer relationship — is that something you'd be open to?

FDE vs. the adjacent titles.

Different companies use different names for roles that overlap heavily with FDE. Here's how to tell them apart.

TITLE	CORE DIFFERENCE FROM FDE
Solutions Architect	Designs the system and hands off a document — typically doesn't write or own the production code themselves.
Sales Engineer	Supports the sales motion technically, but doesn't usually build or ship production systems.
Technical Delivery / Implementation Engineer	Configures and deploys existing product features — less scoping and architecture ownership than FDE.
Consultant	Produces a deliverable and exits. An FDE stays accountable for the system after it ships.
Forward Deployed Engineer	Embeds with the customer, scopes the problem, builds and ships the code, and stays accountable when it breaks — end to end, same person, no handoffs.

> **The one-line test:** if the person doesn't write and stay accountable for production code, the role isn't really FDE — whatever the title says.

If you're targeting AI-native FDE roles.

The core discipline above is track-agnostic. If you're aiming specifically at OpenAI/Anthropic-style or AI-startup FDE roles, add these reps.

▸ ADD THESE EXERCISES

- [ ] Build a small eval harness for any LLM-powered feature: write 20–30 realistic test inputs, score the outputs, and find where the system breaks — this is testing behavior, not just logic.
- [ ] Practice debugging a live model/agent failure with a "customer" watching — same drill as Week 8, but on an AI system instead of traditional code.
- [ ] Learn the vocabulary: context windows, retrieval, grounding, hallucination, prompt vs. context engineering — you'll be expected to use these precisely, not loosely.

> This module is additive, not a replacement. The traits from Phase 1 — ambiguity tolerance, ownership, domain immersion — are what get you hired. This module is what gets you fluent in the specific flavor of problems you'll own.

— FIELD NOTES

YOU DIDN'T LEARN A ROLE. YOU REBUILT HOW YOU WORK.

that part doesn't go away after week 12...

youtube.com/@beyondstaging · keep this for the next 90 days too