


# INTERVIEW PREP

## Atlas Capital

*Senior Staff Engineer, Payments Platform*

Prepared for Sam Lobster

May 2026 — Comprehensive Edition

 This document goes deep. Read it section by section over 2-3 sessions — don't cram. The goal is to internalize the stories and frameworks so you can speak naturally, not recite.

# Company Snapshot

Atlas Capital is a Series D fintech building the unified payments layer for B2B SaaS. ACH, card, wire, and FX in one API. Operates as the money-movement backbone for ~120 SaaS partners; processed \$8B GTV in 2025. Engineering org is ~70 (NY HQ + remote-first), with the payments-platform team being the core technical surface — that's the seat you're interviewing for.

**Stage / Funding:** Series D · \$340M raised total · Last round \$120M Q1 2026 led by Lightspeed at \$2.1B post-money · Prior investors: Sequoia, Founders Fund, GV.

**Scale:** \$8B GTV processed in 2025 · 12-country FX coverage · ~70 engineers · ~120 enterprise SaaS partners · 99.99% settlement uptime SLA

## Key People

- **Maya Park — VP Engineering (your skip-level)** — Ex-Stripe Treasury (2018-2022, IC4 → IC5). Ex-Square Cash App settlement (2015-2018). MS CS Stanford. Strange Loop 2024 talk "Idempotency at the bank-rail boundary" is required pre-reading. Strong opinions on event-sourcing vs CQRS — don't pitch CQRS.
- **David Liu — Director of Engineering** — Ex-Plaid Risk Engineering (2019-2024). Published "Probabilistic Reconciliation in Distributed Ledgers" (SIGMOD 2023, cited 47x). Will probe the probabilistic angle of your work — be ready to defend deterministic reconciliation as the right tradeoff at FintechCo's scale.
- **Priya Khanna — Engineering Manager (your direct manager)** — Internal hire from Atlas operations (2023-present). No external eng-leadership history. Will likely focus on culture-fit + the management dimension rather than technical depth. Lead with team-building stories, not architecture stories.
- **Kelechi Okonkwo — Staff Engineer (peer on the panel)** — Ex-Wise. Owns the partner-bank integration layer today. He may probe whether you'll respect existing architectural decisions or come in trying to rebuild — make it clear you'd start with deep listening before redesigning.

## 3. Detailed Question & Answer Guide

*Each answer includes: the question, what they're REALLY asking, your full STAR answer with specific details, the Atlas Capital tie-in, and common follow-ups with responses.*

### Q1: “Tell me about the most complex migration you've ever led. Walk me through it end-to-end.”

**Story:** FintechCo reconciliation pipeline rebuild

#### SITUATION

At FintechCo (Mar 2023-present), I owned the payments reconciliation pipeline for \$4B/yr in processed volume. The legacy architecture was Postgres + cron — settlement ran nightly, the finance team manually intervened on every discrepancy, and our SLA with merchants was 24h post-clearing. The board was pushing for sub-90-min settlement to enable T+0 payouts as a product wedge.

#### TASK

Cut P95 settlement latency by 90%+ without breaking idempotency or the audit trail finance + our PCI auditor depended on. And I had to do it without disrupting the 2-engineer team's ability to ship features.

#### ACTION

Three moves. (1) Migrated the ledger from Postgres + cron to Kafka topics with an event-sourced ledger model — every state change is an immutable event, settlement is a deterministic projection. (2) Built the settlement-diff tool that catches discrepancies between processor and ledger BEFORE close, so finance's role shifts from "investigate every miss" to "sign off on a clean diff." (3) Co-authored the close-acceleration RFC with the FintechCo CFO so the implementation order matched the business priority order, not just the engineering one.

#### RESULT

Settlement latency 24h → 70 minutes P95. Eliminated 3 of 4 manual close steps that required finance intervention (~28 person-hours/month saved on the finance side). Zero audit findings on the migration. The team grew from 2 to 6 engineers during the project — they were the ones who shipped the second half.

#### TIE TO ATLAS CAPITAL:

The Atlas product probably topped out at the same architectural boundary FintechCo did — the moment when settlement latency stops being a tuning problem and starts being a topology problem. That's the migration I want to lead next.

## Q2: “Whiteboard scenario: Atlas processes payments in 12 currencies via 4 partner banks. One bank's FX feed goes stale at 14:00 UTC. Settlement runs at 17:00 UTC. Walk me through what your system does.”

**Story:** Stale-FX-rate degraded-mode design

### SITUATION

The question is testing whether I can reason about partial-outage behavior when one of the load-bearing inputs (the FX feed) is unreliable but the deadline (17:00 settlement) is fixed.

### TASK

Design the decision tree for what the system does between 14:00 (feed goes stale) and 17:00 (settlement deadline).

### ACTION

Three modes, picked by escalation order. (1) Use last-known-good rates with a stale-rate flag in the ledger; settlement runs on time but every transaction tagged as stale-rate-priced gets a finance-team review row. (2) If the staleness exceeds a configurable threshold (say 6 hours), automatically delay settlement by 1 hour and re-attempt feed. Notify finance + the CFO via the same on-call surface as PCI incidents. (3) If degraded-mode runs hot (more than X% of daily volume across stale-rate-priced transactions), fail open to manual settlement with an explicit human approval gate. Crucially: every mode writes an immutable mode-switch event into the ledger, so finance + the auditor can reconstruct exactly when we were in each state.

### RESULT

I wouldn't ship this to production without first validating the threshold values against 18 months of historical FX-feed reliability data — the answer is heuristic, not derived. But the topology is right: degraded modes that are inspectable by finance, not silent fallbacks.

### TIE TO ATLAS CAPITAL:

I'd want to see Atlas's FX-feed reliability distribution before fixing the thresholds. Decisions like "delay vs use stale" are statistical, not architectural.

## Q3: “Maya's blog post argues for event-sourcing but explicitly NOT CQRS. Why do you think she made that call, and would you have made the same one?”

**Story:** Event-sourcing without CQRS

### SITUATION

Maya's post argues that for a payments ledger at Atlas's scale (~\$8B GTV), the read-path is dominated by point queries (lookups by transaction\_id) and short scans (last 30 days for a customer). It's NOT dominated by complex analytical projections. CQRS adds operational complexity (separate read store, eventual

consistency, double-writes) for a workload that doesn't need the read-path optimization CQRS exists to provide.

## TASK

Articulate why I agree, and where I'd push back.

## ACTION

I agree on the core call. CQRS is a load-shedding pattern — you reach for it when the read shape diverges enough from the write shape that maintaining a unified store costs more than running two. At \$8B GTV with mostly transactional reads, the divergence isn't there yet. Where I'd push back gently: I'd want to know what happens when a finance-analytics workload starts ramping (it usually does — board reporting, reconciliation dashboards, FX-risk modeling). At some volume that read-path DOES diverge. The choice isn't "never CQRS" — it's "not yet." I'd want a clear architectural marker for when we revisit.

## RESULT

I'd ask Maya whether the team has agreed on the trigger condition that would force the CQRS conversation. If yes, that tells me the team is mature about architectural debt. If no, that's the first RFC I'd write.

## TIE TO ATLAS CAPITAL:

The interesting half of architectural decisions isn't "what we picked" — it's "what would force us to revisit." That's where the durability of the choice comes from.

## Q4: “How did you grow your FintechCo team from 2 to 6? Walk me through the hiring rubric and what you learned.”

**Story:** FintechCo payments infra hiring 2024-2025

## SITUATION

FintechCo payments infra was 2 engineers (me + one) in early 2024. Target: 6 engineers by Q4 2025. We had a 17-month timeline and a senior-leaning role mix.

## TASK

Build a hiring pipeline + leveling rubric that produced consistent senior+ engineering hires without diluting the team's existing technical bar.

## ACTION

Three-part system. (1) Wrote the FintechCo eng leveling rubric (Senior → Staff) in collaboration with the VP Eng — the rubric is still the version they use. It's behavior-based, not deliverable-based: "do they design RFCs that other senior engineers reach for as references?" (2) Built a take-home that's an actual production-shaped problem (event-sourced ledger replay correctness) rather than algorithmic puzzles. Took candidates 2-4 hours; I read every submission personally for the first 12 months. (3) Made the on-site a 3-round panel: 1 architecture deep-dive (me), 1 collaboration round (peer engineer), 1 culture fit (VP Eng). No coding round — the take-home covers it. Faster, more honest signal.

## RESULT

Hired 4 engineers in 18 months from 200+ inbound applicants. 4 senior promotions internally on the team I built (we leveled people in alongside the new hires). Zero attrition in 18 months. The take-home format reduced hiring-funnel cost by ~40% vs the previous on-site-coding format.

## TIE TO ATLAS CAPITAL:

The Atlas role is 4 → 8 over 12 months — basically the same shape as the FintechCo problem at slightly higher cadence. The biggest delta I'd want to understand: how does Atlas source senior+ engineers today? At FintechCo I had to build the inbound funnel from scratch.

## Q5: “How do you split your time between IC architect work and the people-management half of this role?”

**Story:** FintechCo operating model

## SITUATION

At FintechCo I was both technical-lead-IC and hiring-manager for the payments-infra team. The naive split is 50/50; the actual right split varies by quarter.

## TASK

Articulate the operating model I actually used.

## ACTION

Three modes, depending on team state. (1) Hiring-heavy quarters (Q1, Q3 each year): 70% management — sourcing, interviewing, onboarding. 30% IC, focused on RFCs and architecture review (the stuff that compounds). I do NOT pick up implementation work in these quarters. (2) Stable quarters (no active hiring): 30% management, 70% IC. I take on one major architectural surface per quarter and ship it. (3) Crisis or migration quarters: 10% management, 90% IC. The team needs me unblocking technical decisions, not managing performance reviews. The performance review cycle gets explicitly deferred.

## RESULT

The model worked because it was EXPLICIT. The team knew which mode we were in each quarter. Promotion conversations didn't sneak up on me — they were calendared into stable quarters. And when migration crunches hit, I had pre-existing permission from my manager to defer the management work.

## TIE TO ATLAS CAPITAL:

The Atlas role probably has Q1 + Q2 in hiring-heavy mode (you want to add 4 engineers in 12 months) and Q3 + Q4 in build-heavy mode. That's the operating model I'd propose to Priya in week 1.

## **Q6: “It's six months in. Settlement breaks at 02:00 UTC, \$200K customer impact, and the on-call engineer is the one who joined three weeks ago. What do you do?”**

**Story:** On-call escalation hypothetical

### **SITUATION**

I'm woken up at 02:00. New engineer is on-call. Active customer impact, growing.

### **TASK**

Resolve the incident, protect the new engineer's confidence, and produce a post-mortem that doesn't blame anyone.

### **ACTION**

I join the call within 5 minutes. I ask the new engineer to STAY ON IT — they walk me through what they see. I don't take over the keyboard. I help them by asking specific diagnostic questions ("have we checked the partner-bank feed timestamp?", "what does the ledger projection lag look like?"). I'm a peer, not a savior. If we need a senior intervention I escalate explicitly to the next senior engineer — not by silently taking over. We resolve the immediate impact within 60 minutes. The post-mortem is co-authored by the new engineer + me. The post-mortem framing is "the on-call rotation got an unfamiliar incident shape during a low-staffing window" — system framing, not person framing.

### **RESULT**

Two outcomes I'd defend. (1) Customer impact bounded; finance team gets a clean reconciliation. (2) The new engineer's confidence isn't shattered — they're more competent next time, not less. If I'd taken over the keyboard at 02:05, neither outcome would have been as good.

### **TIE TO ATLAS CAPITAL:**

The reason this matters at Atlas: you're growing 4 → 8. Half the team in 12 months will be "the engineer who joined three weeks ago." The on-call posture has to be one that doesn't break them.

## 4. Questions to Ask — Detailed Strategy

### Role & expectations

*“What does the first 90 days look like for the person stepping into this seat?”*

**Why this works:** Forces a specific answer; reveals whether they have a real plan or just a job description.

*“What does Priya care about most in the IC-architect-vs-hiring-manager split?”*

**Why this works:** Shows you understand the seat's core tension and want alignment with the actual hiring manager.

### Architecture & strategy

*“Maya's post argues for event-sourcing without CQRS. As volume grows, what's your trigger condition that would force the CQRS conversation?”*

**Why this works:** Demonstrates you read the post AND that you think architecturally about debt thresholds, not just current state.

*“What's the team's hardest unsolved problem this quarter — the RFC nobody's had time to write?”*

**Why this works:** Surfaces the real backlog, not the marketing version. Their answer tells you what your first month would actually look like.

### Team & culture

*“How does Atlas source senior+ engineering candidates today? Is the inbound funnel working, or is hiring this seat partly about rebuilding it?”*

**Why this works:** Half the role is hiring. If their funnel is broken, you need to know before signing.

*“Where's the seam between the payments-platform team and Kelechi's partner-bank integration team? How do you decide who owns what?”*

**Why this works:** Names a peer (Kelechi) — shows you've done your homework. Probes whether the team boundaries are clean or contested.



## 5. Salary & Compensation Strategy

### Posted Ranges

Location	Base Salary Range
Posted range	\$230K – \$290K base · 0.05–0.12% equity · 15% target bonus
Your anchor	\$275K base · 0.10% equity · 15% bonus · \$25K sign-on

**Your ideal ask:** \$275K base, 0.10% equity, 15% target bonus, \$25K sign-on. The sign-on bridges any comp-gap from your current FintechCo seat without making the base number the anchor.

### Leverage Points

- Direct match on every must-have. The two gaps (SOX, FX) are nice-to-haves, not requirements.
- 11 years in role; \$4B/yr ledger ownership; multi-year audit-clean operating record. You're operating at the band's top, not the midpoint.
- IC-architect AND hiring-manager dual scope is rare. Most candidates can do one, not both. Price accordingly.
- Series D close in Q1 2026 — the company has runway. They're not negotiating from scarcity.

### Negotiation Scripts

**When negotiating the offer:**

*"Based on the scope I've been operating at — \$4B/yr ledger, multi-year audit-clean operating record, hiring track from 2 to 6 — and given the role's IC-architect AND hiring-manager dual scope, I'd target \$275K base with 0.10% equity and a sign-on to bridge the move. I'm flexible on the mix between base, equity, and sign-on if there's a structural reason to lean one way."*

## 7. Red Flag Checklist — Things to Confirm

*Use the interview to verify these items. Any red flags should be addressed before accepting an offer.*

- ☐ **No public-company / SOX exposure**

Listed as nice-to-have, not required, but probable interview probe. Inoculate proactively in the opener: "I haven't run SOX. PCI + SOC 2 fluency, plus close-discipline at series-stage scale."

- ☐ **FX / multi-currency settlement gap**

Listed as nice-to-have. Frame as "fastest-to-learn given my reconciliation depth" but be honest about the gap. Don't bluff.

- ☐ **Resume reads more IC than people-manager in years 1-6**

The early-career years don't carry the hiring-manager signal. Foreground the FintechCo 2→6 hiring track explicitly in the opener — don't bury it under technical achievements.

- ☐ **11 years and no Mercury / Stripe / Square brand**

The panel may implicitly probe whether you've operated at the bar these companies set. Use the FintechCo audit-clean record + the 4 internal senior promotions as the implicit benchmark — show outcomes, not logos.

- ☐ **Strong technical match on must-haves**

Every must-have requirement maps to a real shipped story. This is a confirmed strength — don't over-defend it.