Request for Teammates

FOSSA Engineering, Q4 2018

Introduction

This document outlines categories of engineering challenges on our team. During your technical interview, please come prepared to have an open-ended conversation about some of these topics and share your opinions. That doesn't mean you need to have solutions to all (or any) of these challenges. The goals here are:

- To learn more about any relevant domain-specific experience you have with our challenges (it's okay if you don't have any!)
- To learn more about which of these challenges interests you
- To answer any questions you have about what our engineering team is working on

Categories

From prototype to production-ready

All startups begin by building prototypes, starting with an MVP to raise money and moving to an SRP to close the first sale. This engineering process qualitatively changes when you begin building real customer traction. Now, we need to transition from “move fast and break things” to “move fast with stability”.

- How do we prioritize refactoring against the need to iterate on features?
- When, if ever, do we make major architecture changes? What factors go into that decision? What common changes have you seen, and how have they worked out?
- Have you ever done a rewrite? A language migration? A major database schema migration? How have those worked?
- Have you lived the MVP-to-production process? What are your insights on this process?

Building excellent engineering teams

Having a transparent, safe and candid culture is extremely important to our team. We want to cultivate a trusting environment where team members feel like they can do their best work, and build a process for growing our team that is good for both us and our candidates.

- How do engineering teams evolve as they grow larger? As the business grows larger?
- When does it make sense to add process to engineering and project management? How do we do that correctly, and evaluate how effective our process is?
- How do we discover, attract, and recruit great engineers?

- How do we maintain a culture of openness, transparency, and safety as our team grows?

Designing great developer experiences

We integrate with deeply technical developer tools. Build and dependency management systems are often notorious for obscure failure modes, unhelpful error messages, and complex operation. Despite this, we are committed to shipping an excellent product. This means we need to build intuitive developer experiences on top of highly complex systems.

- What are the best (and worst) developer tools you've worked with?
- What makes a great developer experience?
- How do we create tools that are easy to integrate in enterprise settings?

On-premises deployments

On-premises deployments are a painful reality when dealing with enterprise customers. Each on-premises deployment is its own unique beast: it comes with its own set of customer contexts, hardware and software quirks, and access restrictions.

- How do we support and upgrade an on-premises deployment?
- How do we debug a deployment without remote access?
- What are common gotchas with deploying in air-gapped environments?
- How do we do major architectural refactors (e.g. major database schema migrations) while supporting on-premises deployments?

Creating awesome products for enterprise

Traditional enterprise software vendors are driven by their sales team. We believe that our sales should be driven by great product.

- What are your favorite products, brands and design patterns?
- What product blogs or designers do you follow? What are you favorite products to use?
- What are your philosophies behind building delightful user experiences?
- How do we build maintainable, complex front-end applications?
- How do we repair, refactor, and migrate off of unmaintainable front-end code?
- Do you have strong opinions on the state of front-end engineering? (In particular: state management, compile-to-JS languages, testing)
- How does building for enterprise affect front-end design?

Scaling cloud infrastructure

Our small engineering team operates cloud infrastructure with an outsized impact. We solve deeply technical problems with low latency and high reliability. Each time our infrastructure fails, somebody’s build breaks.

- What are common infrastructure scaling pains you've seen? Are they in the runtime? Database? Operational infrastructure (e.g. schedulers, orchestrators, etc.)?
- When is it worth re-architecting a complex system?
- When does it make sense to begin worrying about optimization? Do you have experiencing optimizing PostgreSQL?

Bring your own passion

We love thoughtful people, and hopefully some of our notes above give you a sense on things we're working on. Every team member at this stage contributes to thought leadership at the company. If you have a topic that's not listed here that particularly interests you, we would love to hear about it.