Strangle The Monolith

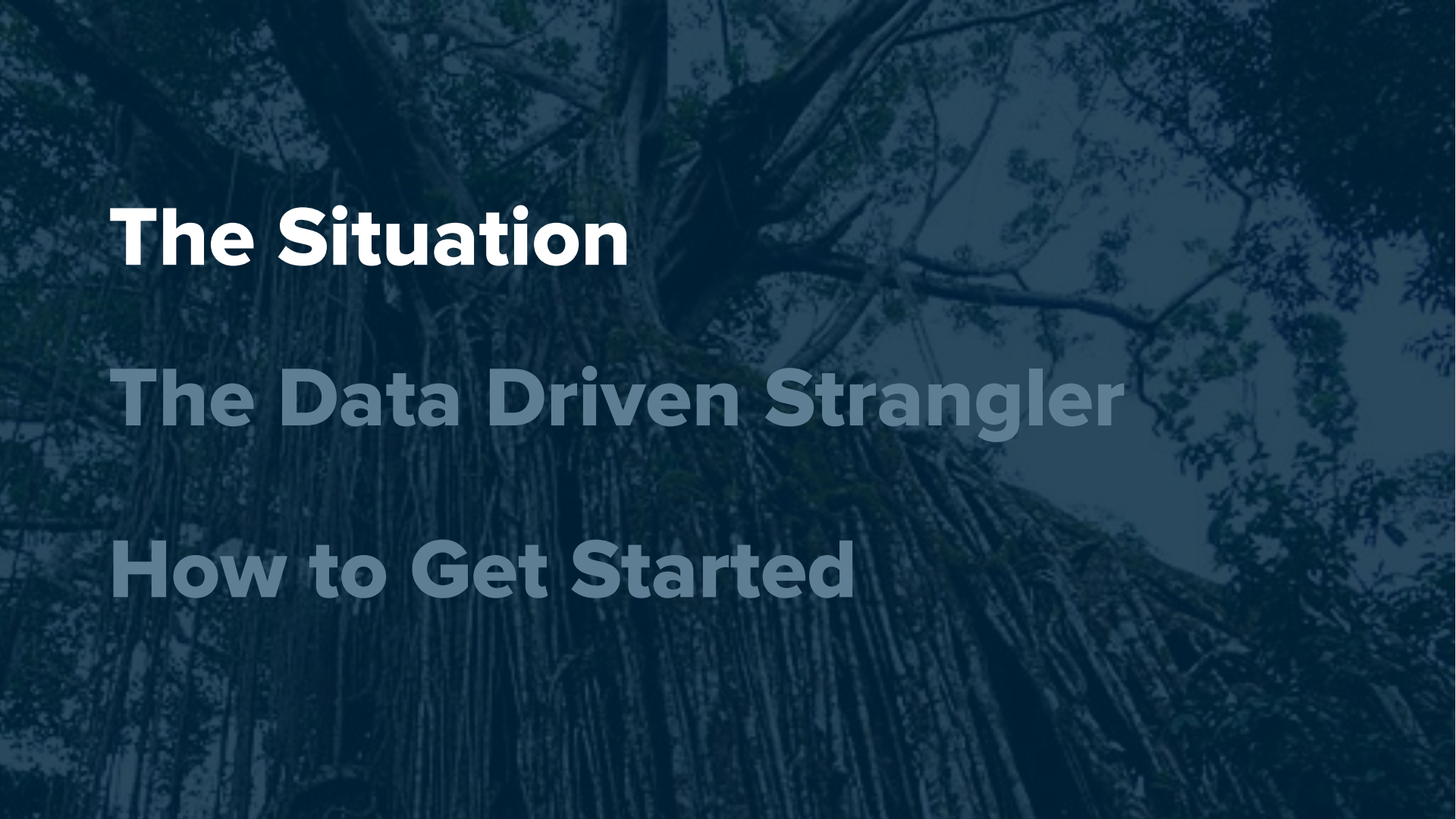
A Data Driven Approach

Amjad Sidqi, Associate Director | **Pivotal Labs**

David Julia, Director | **Pivotal Labs**

HARD TO CHANGE


The Situation
The Data Driven Strangler
How to Get Started


The Situation
The Data Driven Strangler
How to Get Started

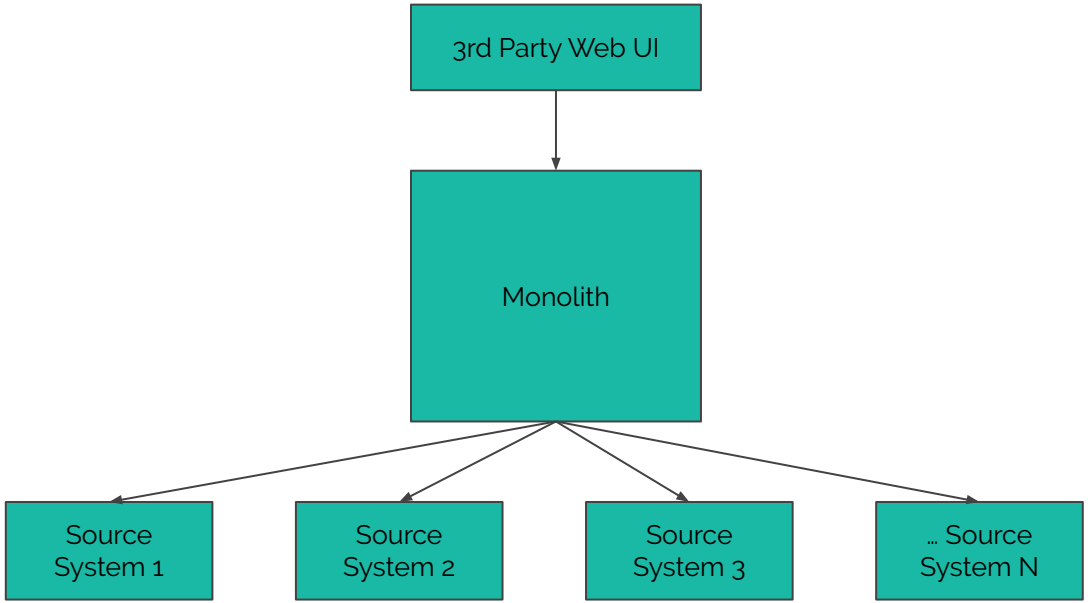
The Scenario

Additional business features

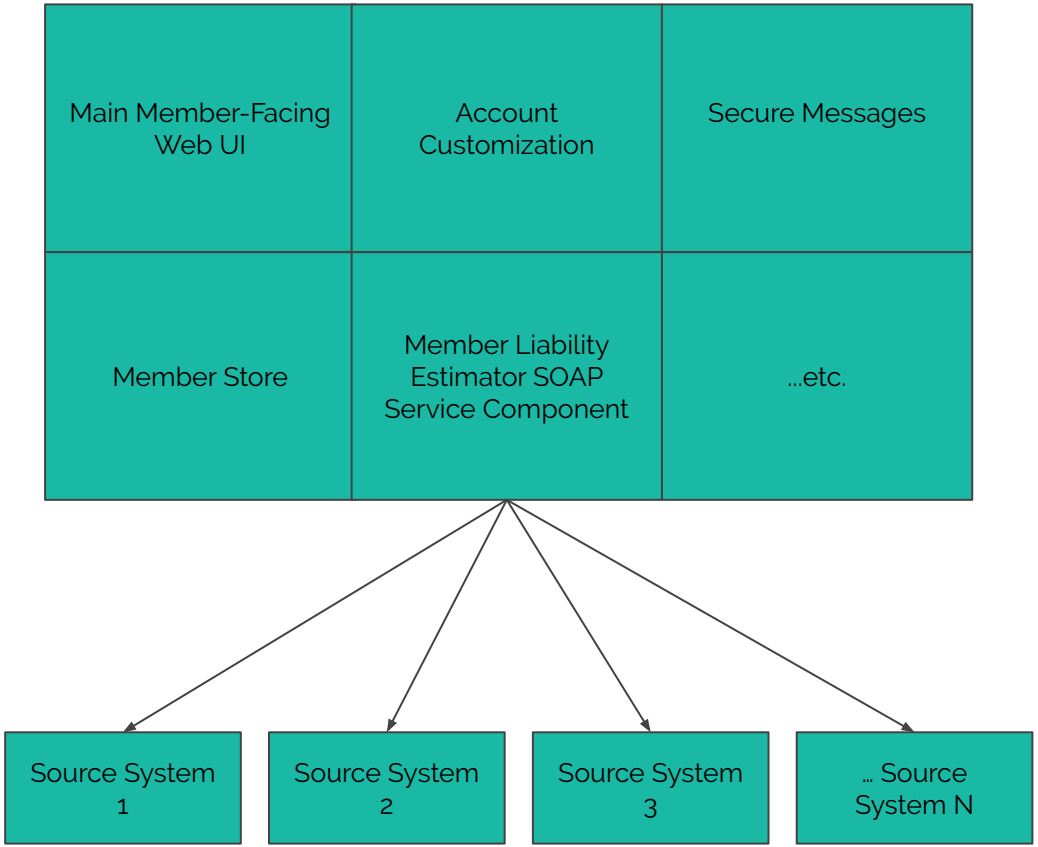
Cost Estimator for medical procedures

Financial Penalties for inaccurate estimations

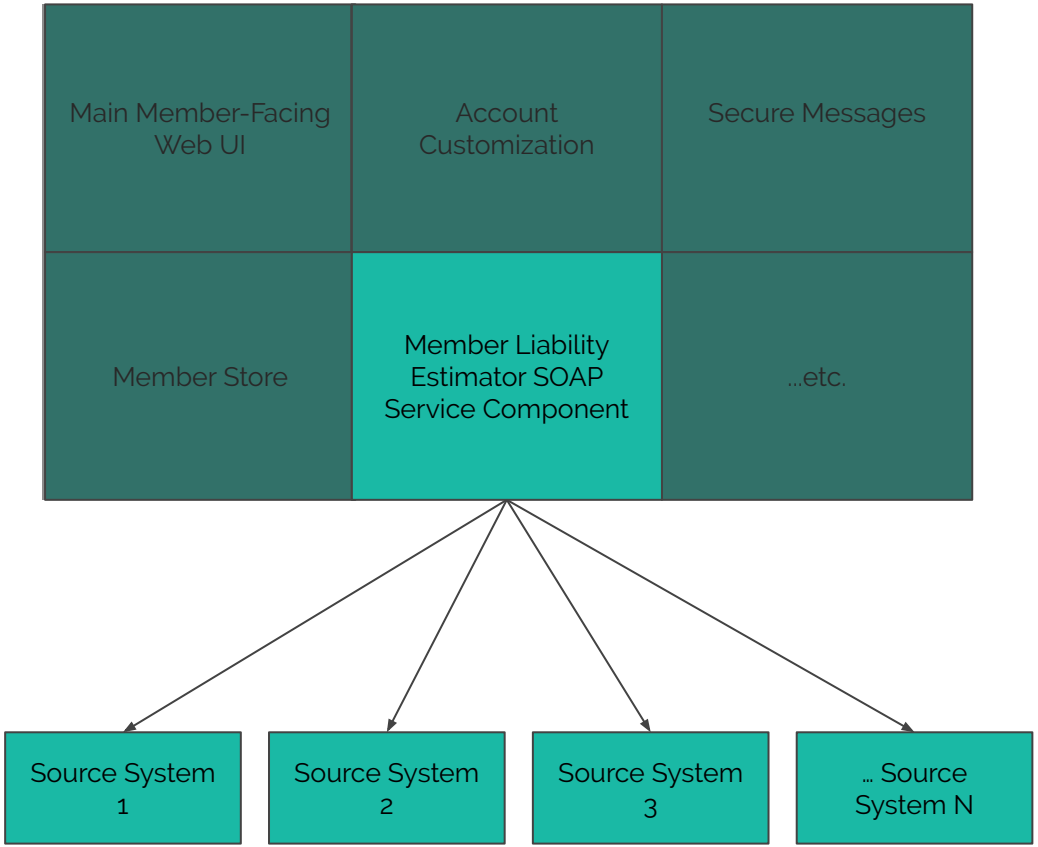
Existing Architecture



Peeking Inside the Monolith



Peeking Inside the Monolith



Commit to the rewrite:

A new API that returns the same results & supports “tiered” networks

Initial approach

The expert leads the way

The Strangler Pattern: An Iterative Rewrite

Benefits of a rewrite with reduced risk, faster time to value

Does require investment in the approach.

Strangler Fig

Hollow Inside of Strangler Fig

Uncertainty

Complex flows create anxiety

Fundamental assumptions were wrong

Pssst... This isn’t working

Strangler is great for decomposition.

BUT

We couldn’t know what logic to build in our new services.

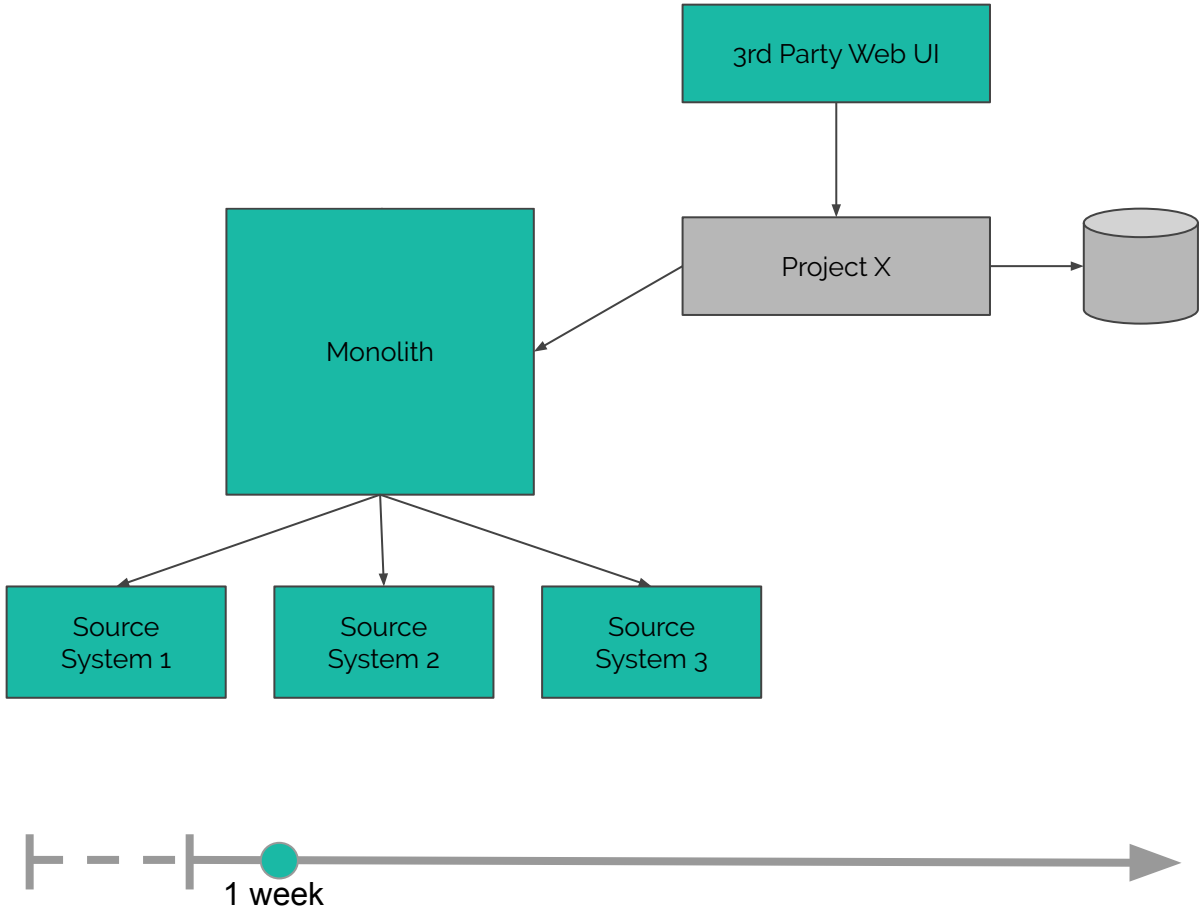

The Situation
The Data Driven Strangler
How to Get Started

Enter The Data Driven Strangler

=

Pass-through & Log (in prod)

Collect Request/Response Data

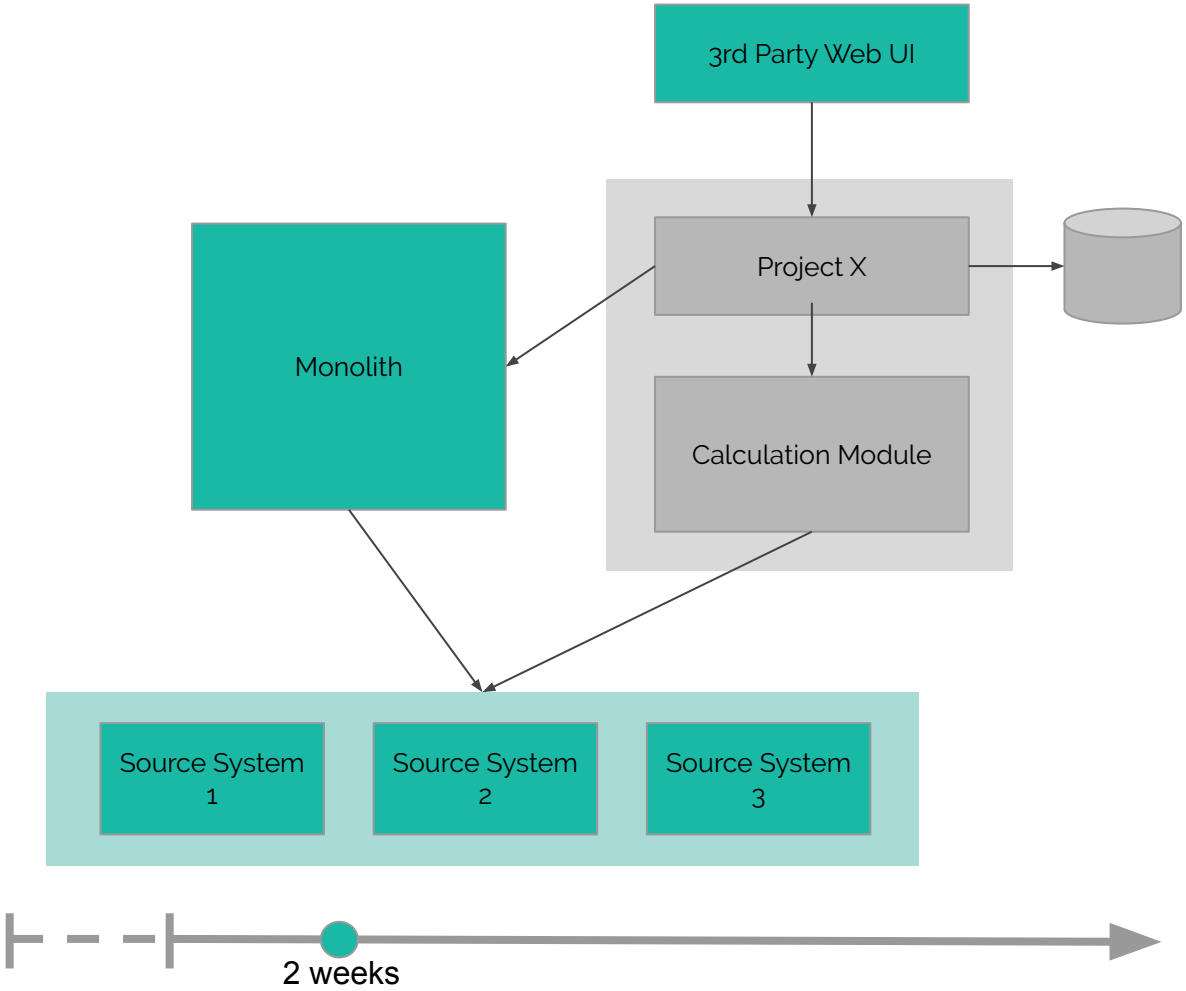


It was like turning on the lights

Log Both Results & Default

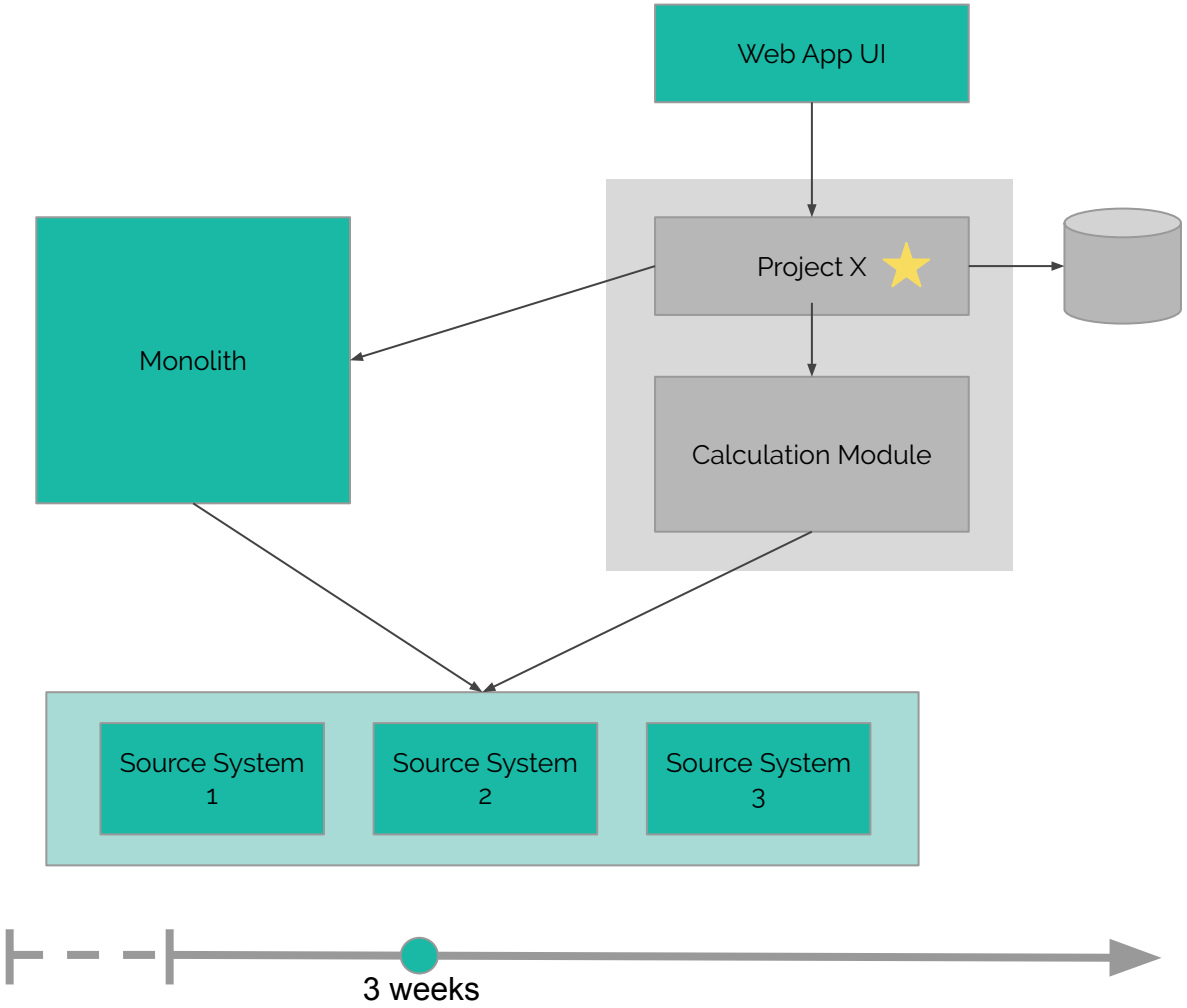
Collect Request/Response Data for Both

Defaulting → No Risk of Bad Result



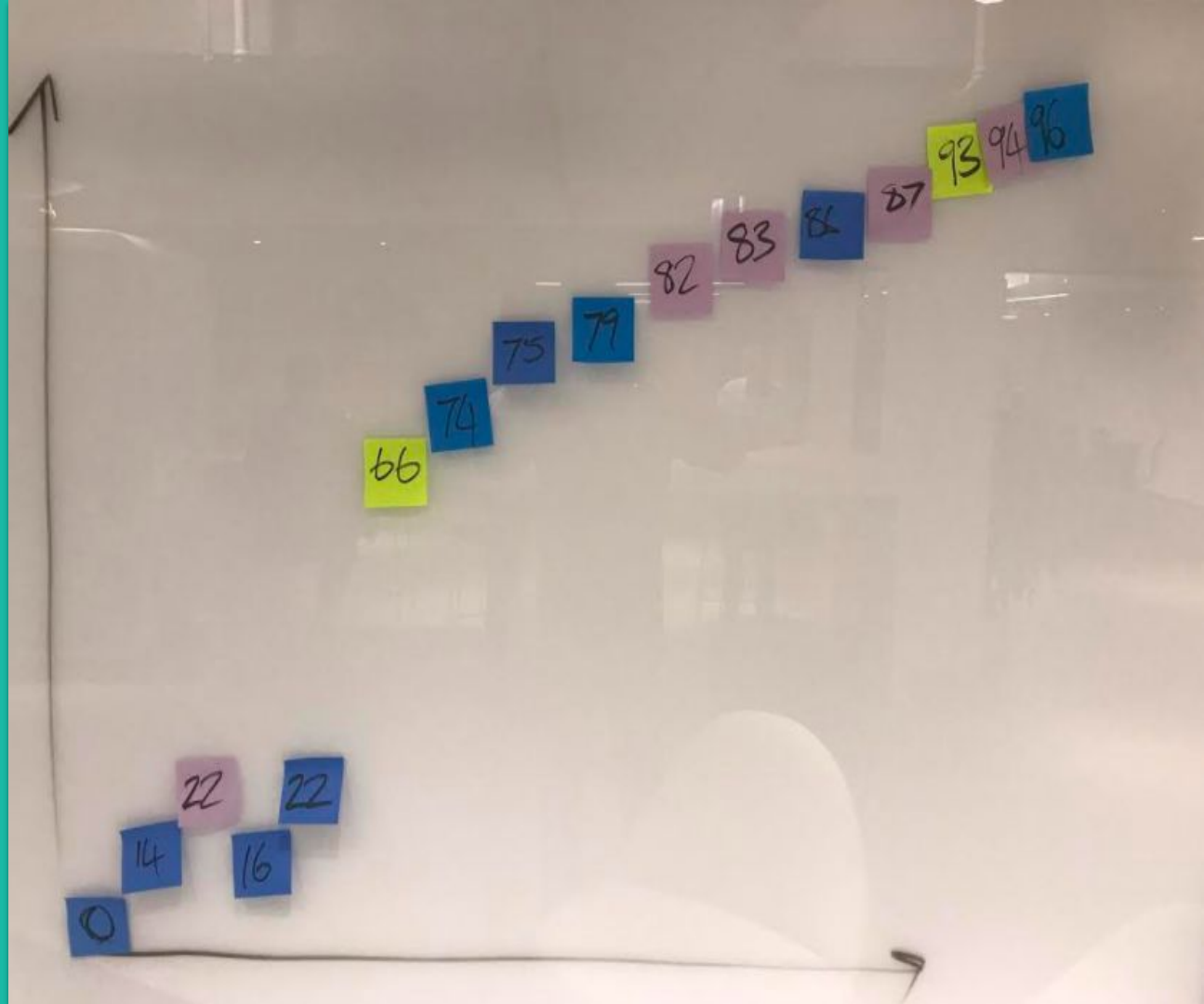
Automate Analysis

Log The Deltas!



We optimized for near real time feedback loops

Let’s focus on what matters ...



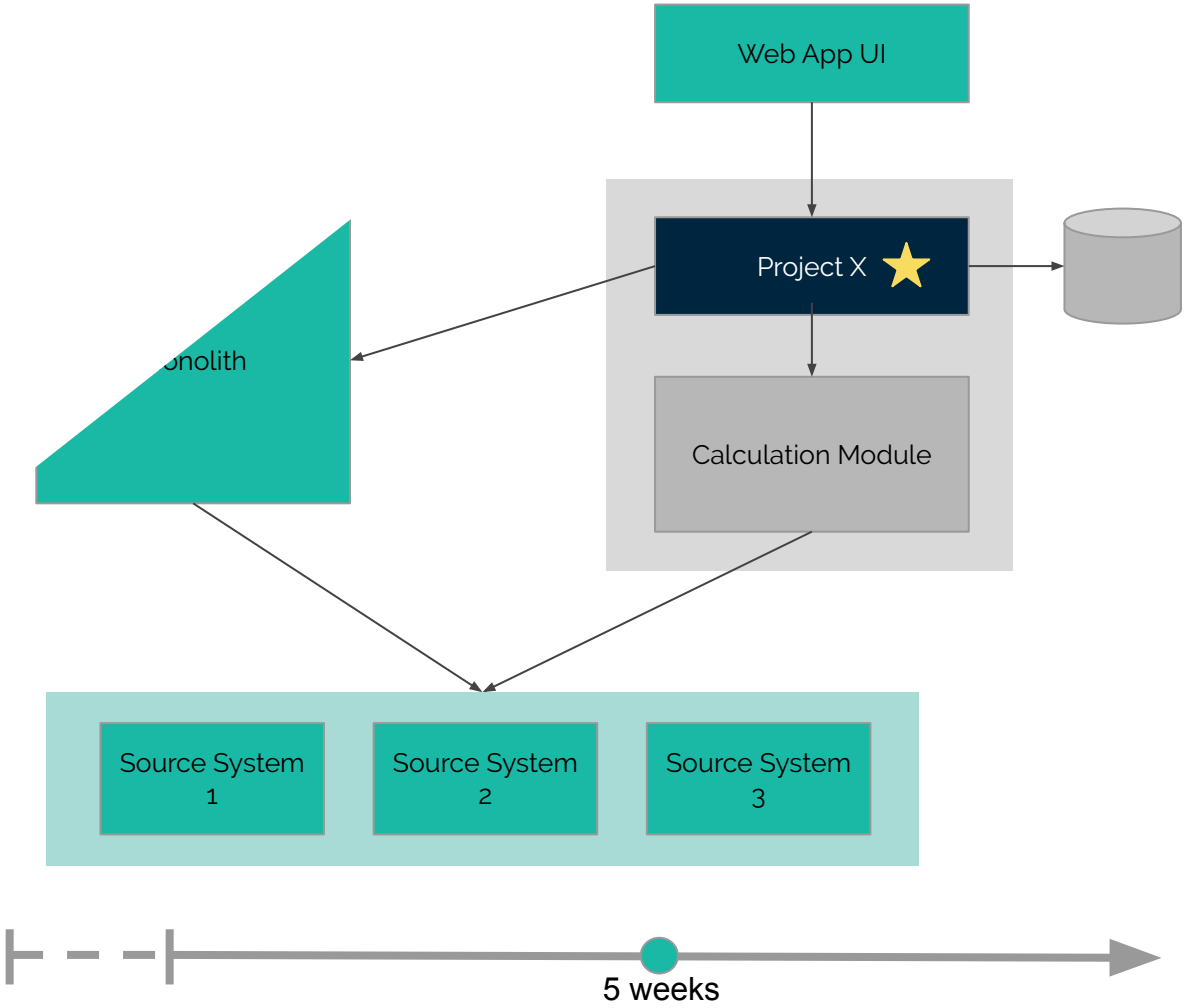
% Error Cases ✕ Avg. $ Diff ✕ Avg. Requests/Day

=

Possible Financial Impact/Day

Started to turn off path to old system for some cases

Starting to strangle stable cases



When

Possible Financial Impact/Day

<

Cost to Maintain Legacy

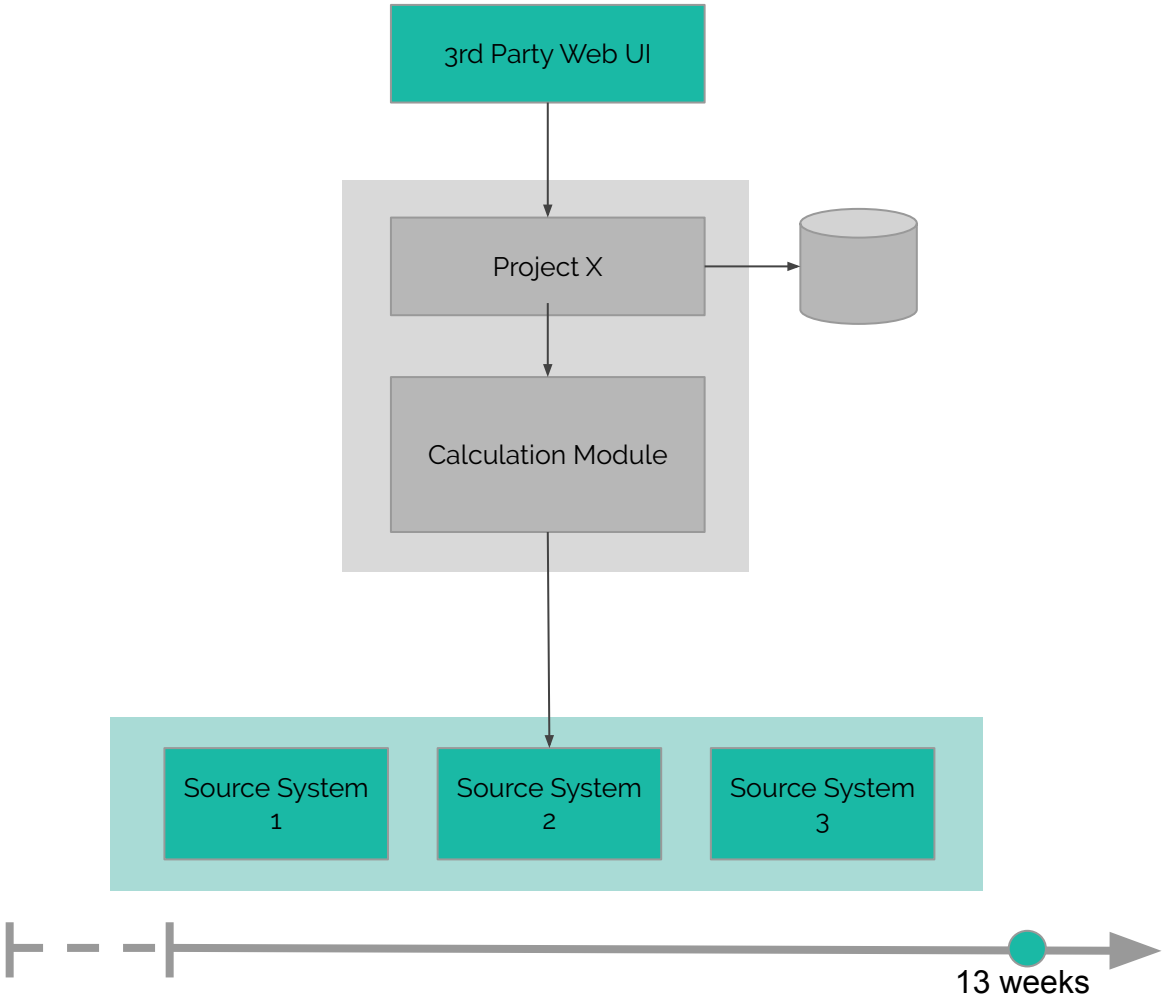
Then...


Shut it down

Shut down the Legacy Calculation Path

Only call into our new calculation module

We've now strangled a large part of the monolith!



This made us feel great!


The Situation
The Data Driven Strangler
How to Get Started

Did any of that sound familiar?

Are you thinking of a rewrite?

Is legacy technology holding you back?

Data-Driven Strangler is Not a Silver Bullet

Options

1. Rewrite from scratch
2. Buy off the shelf
3. Do nothing
4. Containerize
5. Strangler Pattern

Build vs Buy → Build & Buy

Is it core to your business?

Somewhere you want to differentiate?

Will the buy option require a lot of customization--building logic into the system?

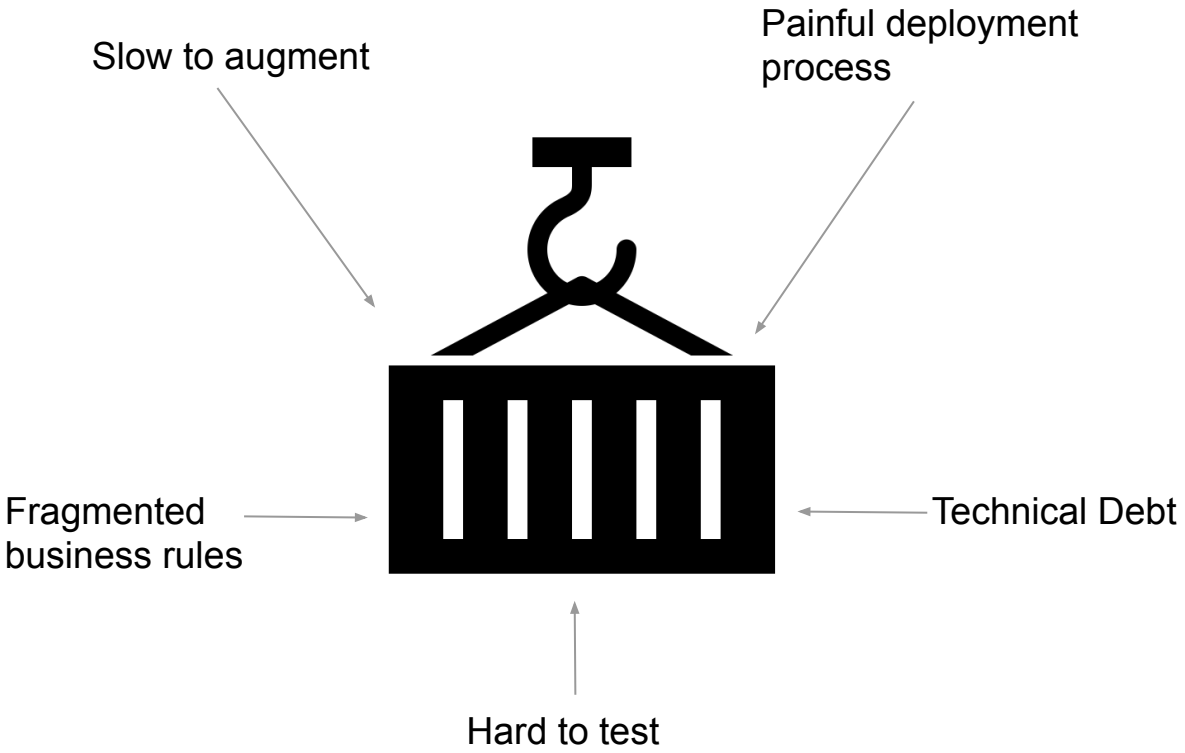
Often, the best option is both: Build the differentiating parts, “buy” commodity components (eg don’t build your own SendGrid, don’t build Stripe, don’t build your own cloud platform).

When to ‘Do nothing’?

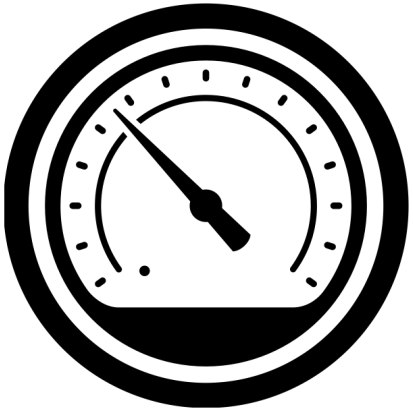
- No delivery pressures
- Low strategic importance
- Stable enough if not touched
- Opex costs under control

What about Containerizing?

Doesn’t actually solve your problem



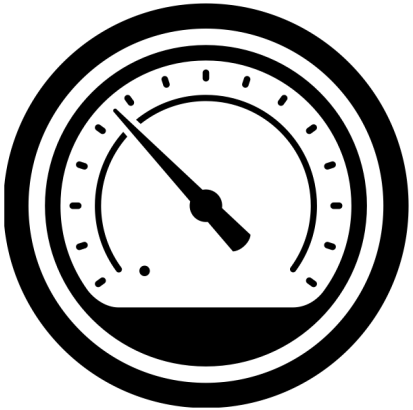
When should you rewrite?



Maturity/Traction of product

- Original product was way off the mark, didn't achieve goals (eg no user adoption).

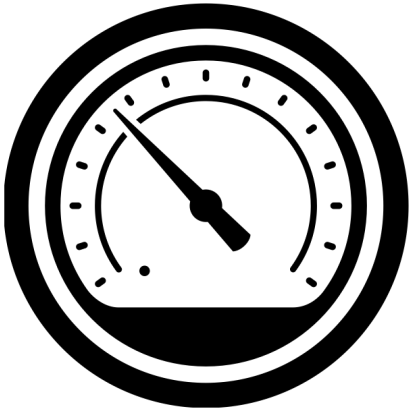
When should you rewrite?



Maturity/Traction of product

- Original product was way off the mark, didn't achieve goals (eg no user adoption).
- Original product does not have traction

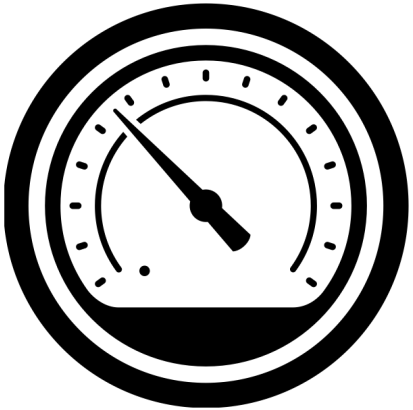
When should you rewrite?



Maturity/Traction of product

- Original product was way off the mark, didn't achieve goals (eg no user adoption).
- Original product does not have traction
- Significant deviation from original intent of product, going after a new market

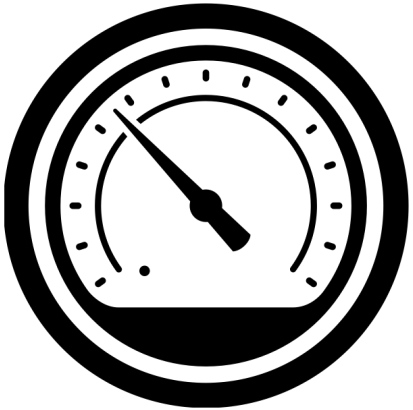
When should you rewrite?



Maturity/Traction of product

- Original product was way off the mark, didn't achieve goals (eg no user adoption).
- Original product does not have traction
- Significant deviation from original intent of product, going after a new market
- Technology holding you back (Mainframe, Visual Basic overly-customized SFDC or AEM)

When should you rewrite?



Maturity/Traction of product

- Original product was way off the mark, didn't achieve goals (eg no user adoption).
- Original product does not have traction
- Significant deviation from original intent of product, going after a new market
- Technology holding you back (Mainframe, Visual Basic overly-customized SFDC or AEM)
- You can redefine the business process around the new system.

When to use the Strangler Pattern

- Well established product with significant user base
- A significant risk to revenue streams
- Lots of necessary complexity in your existing product (eg complex regulatory compliance rules)
- You don’t know the business rules in the existing system

Learnings/Takeaways

- This sounds technical but **don’t compromise User Centred Design**
- An opportunity to **remove complexity**
- Get laser focused on what really matters **80:20**
- **Don’t rebuild like for like**
- When rewriting take an **iterative approach**

How do you do this in your organization?

Start Small

Put together a business case around a *subset* of the capabilities that will deliver value over a matter of months, not years. Frame it as a “no regrets” move with near term benefits.

Quantify Outcomes

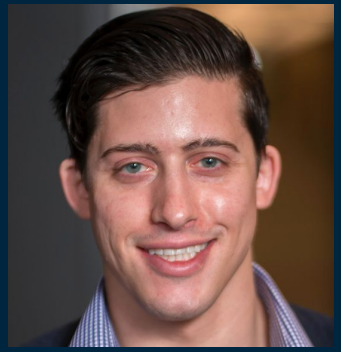
Establish a baseline and measure against it (dev cycle time is good, but cost/revenue/acquisition metrics are even better)

Use one win to build momentum for the next

By starting small, you can prove out the process and build support to keep going. Once you have a first win, a technical foundation, and understanding of the system, you can “double down” and scale the effort.

With the support of Pivotal!

Get in Touch!



email: djulia@pivotal.io
Twitter: @DavidJulia

email: asidqi@pivotal.io

We Love Feedback

What would you like to hear more about?

What questions do you still have?

And...
We are hiring