



Finding Balance

How to build processes that
help not hurt your
engineering team

Two decorative dashed lines in shades of cyan and teal, curving across the bottom half of the slide.

STYTCH

Why is process good?



Improve efficiency by reducing back and forth; minimize downtime and bottlenecks



Ensure prioritization accurately takes into account scope



Clear direction and goals; people want to know what they're working on and what success looks like



Predictability and cross functional transparency; allows other teams that rely on features shipping to plan accordingly

How much process is right for your team? It depends!

How fast is it to ship the average feature?

If shipping and iterating is straightforward for your product, you can probably get away with less process

How clear are your team's goals and priorities?

If prioritization decisions are fast, and there isn't much whiplash, you can probably get away with less process

How big is your team?

Complexity scales with engineering org size, the bigger your org and company is, the more process you likely need

Signs you might need **more** process

How do you know if you need more process?



Lots of time week to week is spent discussing prioritization; More upfront time planning can save valuable time in the long run



Team's are working hard but not shipping at the expected velocity; it's likely that there's lots of stops and starts on work slowing things down



Managing dependencies is difficult, teams that rely on other teams for work have to plan in significant wait times to get capacity from others



Chaotic or unclear priorities, can often feel like the team is just operating like a LIFO queue



Signs you might need **less** process

How do you know if you have too much process?



Engineers are spending more than ~1 hr a week regularly in planning meetings



Planning a project consistently takes longer than executing on it



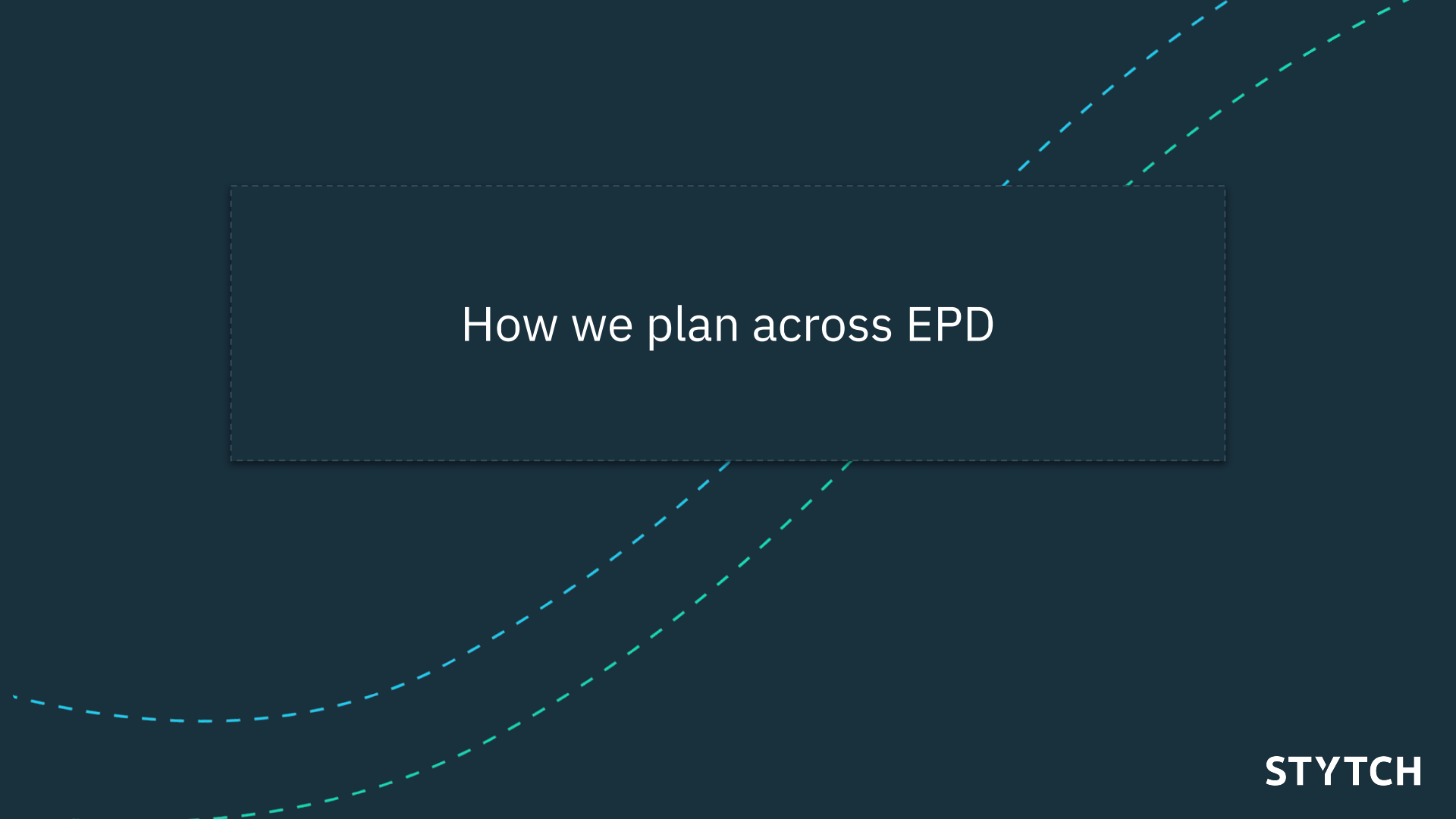
You 100% predict what the roadmap will look like; either you're spending too much time crafting it or there's no flexibility to adapt



Hitting planning metrics is celebrated more than the value that's delivered to customers by shipping those features

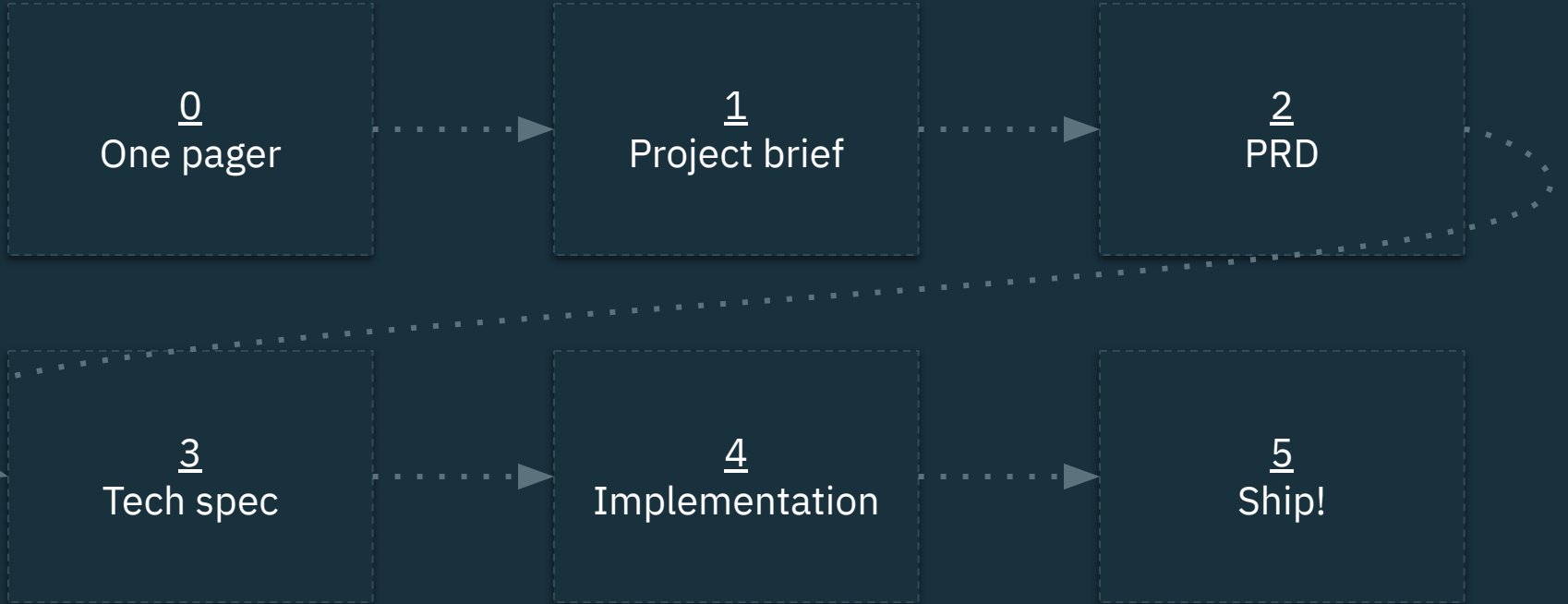


Engineers don't feel empowered to make decisions and need to wrangle many stakeholders to move things forward

The background features several decorative, curved dashed lines in a teal color, creating a sense of motion and depth. A central white text box is positioned in the upper-middle section of the slide.

How we plan across EPD

Lifecycle of a project - a common interface for all projects



Step 0: One pager



One pagers are a helpful tool for exploring the problem space and scoping a project



They are most frequently used during quarterly planning to understand scope for a project so we can tile and draw the cut line for the quarter



Team will typically assign one pagers during the planning process, outside of this, check with your manager if a one pager is necessary or if a project brief will suffice



Post in #open-projects slack

Step 0: One pager

Context

Currently, we do not have pizza at Stytch. This is leading to declining code quality and engineering morale as the lack of pizza is leading to hungry engineers. To correct this, we're going to make pizza.

Goals

1. Make pizza
2. Meet all dietary requirements of engineers
3. Be able to make more pizza in the future

Non-Goals

1. Feed other teams at stytch with pizza
2. Sell pizza to customers

Proposed Solution

Buy equipment needed to make pizza, buy ingredients, and cook pizza.

[Optional] Alternatives Considered

1. Ordering pizza - build vs buy doesn't make sense due to our expectation that we will need more pizza in the future
2. Making bacon-wrapped hot dogs - much more difficult to fulfill dietary requirements

Security / Compliance / Legal Considerations (If applicable)

(Security, Compliance, Legal, and overall risk considerations to be noted for proposed/alternative solutions and or performing no actions.)

Timeline

1. [1 week] Write tech spec
2. [1 weeks] Prep work
 - a. Gather dietary requirements from engineers
 - b. Buy equipment
 - c. Buy ingredients
3. [1 week] Make pizza
 - a. [1 day] Make dough (knead/shape, ferment)
4. [1 day] Bake pizza
 - a. Preheat oven and stone
 - b. Add toppings
 - c. Bake

Stretch: [3 days] Make sourdough starter

Total:

- MVP: 1.5 weeks
- With stretch goals: 2 weeks

Step 1a: Project brief - WIP draft



Project briefs are used to kick off a project and ensure alignment across stakeholders



Project brief is started – typically by Product, but can be anyone who is the DRI for defining the goals of this project, and why it's important



At this point the Project Brief should contain context on the motivation for this project; what are the goals and why is this important to tackle?



Get the ball rolling on spinning up the project team by creating a #project-channel with the XFN group and schedule a kick-off meeting

Step 1b: Project brief



Prior to the kick-off meeting, the XFN group should try and add any constraints, assumptions, open questions that are relevant to their domain area to the brief



During kick-off, ensure everyone has the necessary context on the project and goals – and spend time live discussing any additional questions or helpful context



Align on next steps and owners for discovery/validation, are there any key open questions that should block starting the tech spec?



Post in #open-projects slack

Step 1: Project brief

Motivation

[TO DELETE] This section should provide an overview of any necessary context on:

- *The current state of the world*
- *The customer pain point we are trying to solve or opportunity we are trying to capture*
- *The impact to Stytch's business*

Constraints & Assumptions

[TO DELETE] This section should highlight any assumptions and constraints that will shape the final implementation design – and how we plan on validating these assumptions. This should include:

- *Technical constraints*
- *Customer commitments on product design or delivery timeline*
- *Marketing or PR deadlines*
- *Customer requirements or preferences*
- *Security, Legal, Compliance requirements / concerns*

And a discussion of why we assume this to be true, what the risk is if we are incorrect, and how we are planning to test or validate the constraint/assumption.

Key Open Questions & Trade Offs

[TO DELETE] This section should outline the key open questions or trade-off decisions that we will need to answer before we are able to clearly define a final solution design

Next Steps

[TO DELETE] This should cover the next steps that the team will take in order to answer the open questions, validate any assumptions and constraints, and flush out potential risks

- @-Person is doing X in order to validate Y assumed constraint

Step 2: PRD (product requirements doc)

Product (or acting “product DRI”) is responsible for taking the input from discovery and validation and producing a PRD covering:

- Motivation (which should also be covered in initial project brief)
- Key decisions that might be controversial or informed by constraints/assumptions validated during discovery
- Detailed product requirements to ensure that there is a central place of truth outlining all of the core requirements for this project

Step 2: PRD

tl;dr

Quick summary of the problem we are solving, why it is important to Stytch's strategy and the proposed solution.

Relevant Resources

Here are some other resources you can find related to this project (if they exist). At the initial PRD review stage, most of these won't exist yet but please try to add them as they are created:

- One Pager
- Project Brief
- Engineering Technical Design (Doc link)
- Product Design (Figma link)
- Linear project (Linear link)

Scope

Requirements & Goals

List the high level requirements and goals that are in scope for this project so that stakeholders understand the high level scope before diving into the details

Non-Goals

List explicit areas we do not plan to address.

- Explain why they are not goals
- These are as important and clarifying as the goals

Solution Design

Key Decisions Overview

Call out the most impactful or potentially controversial decisions in this proposal, in order to ensure that all stakeholders can easily review and they won't get lost in the detail of the broader spec.

User Stories

Provide user-centric overview of the core behavior that this project will enable. Please be specific about both the persona and the action and cover all edge cases.

Persona	Action
<persona>	As a <detailed persona> I...

Detailed Product Requirements

Requirement 1

Cover the following:

1. What is the core behavior required
2. Who is impacted
3. What edge cases are involved
4. What other alternatives were considered
5. Why has this path been chosen over the alternatives

Step 3: Tech spec



At this point, the high-level discovery should be done and the project direction should be de-risked. Now we need to figure out the details of how



Use the tech spec template, it can be adapted but helps to ensure certain key components aren't forgotten such as metrics, customer facing changes, etc



Tag people for review and make it clear what you're looking for in their review (i.e. looking for feedback on xyz piece from person a)



Make sure key reviewers approve the tech spec before starting on any controversial pieces of the project



Finally, post in #open-projects slack (yes there have been 3 posts about this project at this point, this helps to ensure alignment throughout each stage)

Step 3: Tech spec

HOW TO USE THIS TEMPLATE:

- Anything in *italics* is a question you might want to consider when designing your product. Feel free to delete the questions once they've been answered.
- MOST parts of this template are optional - if they don't apply or feel like overkill for what you're designing, delete them!

Signoff

Create Linear tickets for each team member you'd like to request a review from.

Team member	Status
Jane Doe	Not Started ▾
John Doe	Not Started ▾

Goals

- 1.

Non-Goals

- 1.

Relevant Links

Is there a PRD or One Pager for this project? Are there public RFCs that we are implementing? Do we have more in-depth documents or Slack threads for certain decisions that have been made?

- Linear Project Link Here
- PRD Link Here
- Github Discussion Here
- Slack Thread Link Here

Step 3: Tech spec

M1: Core API

API Endpoint Change Template

What are the inputs and outputs of the endpoint? Which fields are required, and which ones are optional? You can capture the API schema in abstract, or you can be more explicit with a proto or typescript schema.

UpdateOrgPizzaPreferences – PUT /v1/b2b/pizza/{organization_id}/preferences		
Request		Notes
organization_id	string	Path Param
favorite_topping	Optional string	Could be an enum?
extra_cheese	Optional boolean	
Response		
pizza_preferences	PizzaPreferences	
Considerations		
Billing Tiers	Only Scale and Enterprise customers can set Pizza Preferences	
RBAC permissions	Yes - organization.edit.pizza-settings	
Rate Limiting	X Reqs/Sec by OrgID	

Endpoint Behavior

Describe the endpoint in terms of observable end-user behavior. What are the happy-path flows? What are the edge cases you'll need to address? The goal should be to define the suite of JSON tests and acceptance criteria up front.

- **Happy Path**
 - **When** the pizza preferences are successfully updated, **Then** return a 200
 - **Given** an organization without a favorite topping, **When** the favorite topping is changed to pineapple, **Then** a "Welcome to Team Pineapple" email is sent
- **Failure Cases**
 - **Given** an organization containing a lactose intolerant member, **When** extra cheese is enabled, **Then** return a 400 extra_cheese_forbidden error
- **Implementation Details**
 - Lactose intolerance is cached in Redis, according to the cache proposal described in the [Appendix](#)

Database Schema Change Template

organization_pizza_preferences		
Field	Type	Notes
organization_id	varchar(128)	Primary key, already exists
project_id	varchar(128)	Already exists
favorite_topping	varchar(128)	Can be null
extra_cheese	bool	
secret_ranch_recipe	text	encrypted

Are there certain new query patterns we expect to deal with? What lifecycle methods do we need for this table? Do we have indices for all the expected access patterns?

We'll be adding several new fields to the existing organization_pizza_preferences table.

- **Expected Read Access Patterns**
 - Get by Organization ID - for both single OrgID and Bulk OrgID
 - Search organizations by favorite topping
- **Expected Write Access Patterns**
 - Upsert by Organization ID - for general settings updates
 - Bulk update by Project ID + Favorite Topping - for when toppings are deleted
 - Delete by Organization ID
- **Indexes**
 - (project_id, organization_id) -> used
 - (project_id, favorite_topping) -> used for search, bulk topping delete

Step 3 continued: Tech spec - estimation



Estimation is a key part of the tech spec, to ensure we know what we're getting ourselves into and uncover any potential gotchas



1 point == 1 hour of **ideal time** (not actual time)



Estimation should be a tool, not something to stress over. The goal is to spend just enough time estimating to ensure predictability but not over planning

Step 3 continued: Tech spec - estimation

Timeline

When estimating tasks, make sure to use ideal time (concentrated, uninterrupted, focus time) as opposed to actual time (includes shoulder taps, interviewing, meetings, bugs, fires). Here's a [link](#) for more information on how we estimate.

TL;DR: 1 point = 1 hour of focus time

Does the project require changes to other codebases - like the docs, dashboard, event log, or SDKs? Do we want to set aside extra time for manual testing or bug bashing?

Milestone 1

1. [16 points] Task 1
2. [8 points] Task 2
3. [4 points] Task 3

Total: 28 points

Milestone 2

1. [4 points] Task 1
2. [8 points] Task 2
3. [4 points] Task 3

Total: 20 points

Engineer 1: 20 points / week

Engineer 2: 8 points / week

Milestone 1 estimate: 1 week

Milestone 2 estimate: 1 week

Step 4: Implementation



Create a linear project and add the tickets you sketched out in the tech spec



Decide how you want to run the project day to day, often teams do live or async standups depending on the project and team involved



Ensure project updates from linear are hooked up to #updates-epd

Step 4: Implementation

The screenshot shows a project management interface for a project named "Make Pizza". The interface is divided into several sections:

- Header:** "Projects > Make Pizza" with navigation icons for Overview, Issues, and a search icon.
- Project Card:** Features a pizza icon, the title "Make Pizza", and a description: "This is an example project that demonstrates how to plan a project."
- Properties:** Shows status "In Progress", priority "No priority", category "Developer Relations", and assignee "julianna". It also includes a date range "Aug 1 → Sep 5".
- Docs & links:** Contains a link to "Productlane Feedback".
- Description:** States "We'll be making pizza for the eng team to ensure engineers can stay fueled."
- Milestones:** Lists three milestones: "Gather ingredients", "Make dough", and "Bake pizza", each with a right-pointing arrow. A "+ Milestone" button is at the bottom.
- Updates Panel:** Located on the right, it shows a green "On track" status, a text input field "Write a project update...", and a calendar view for "September 2024". A specific update is shown: "On track by julianna Sep 5" with the text "Pizza making is going well! We've done initial research on requirements and are ready to start on milestone 1 next week." and interaction icons for comments and reactions.
- Activity:** A log entry at the bottom right states "julianna created the project · Sep 5".

Step 5: Ship ship ship



Develop a roll out process, this often involves feature flags and a bug bash where Stytc employees try and break a new feature



Work with Developer Success for any beta customers that might want to get early access to this feature



Post in #all-launches so everyone can celebrate the launch and go to market teams have visibility so that they can share the new feature appropriately



Why this works for us

What works for our team



Engineers own execution on projects: project leads are responsible for driving the tech spec, breaking it down into milestones and tickets, and driving execution week to week. Managers hold engineers accountable but give them space to execute.



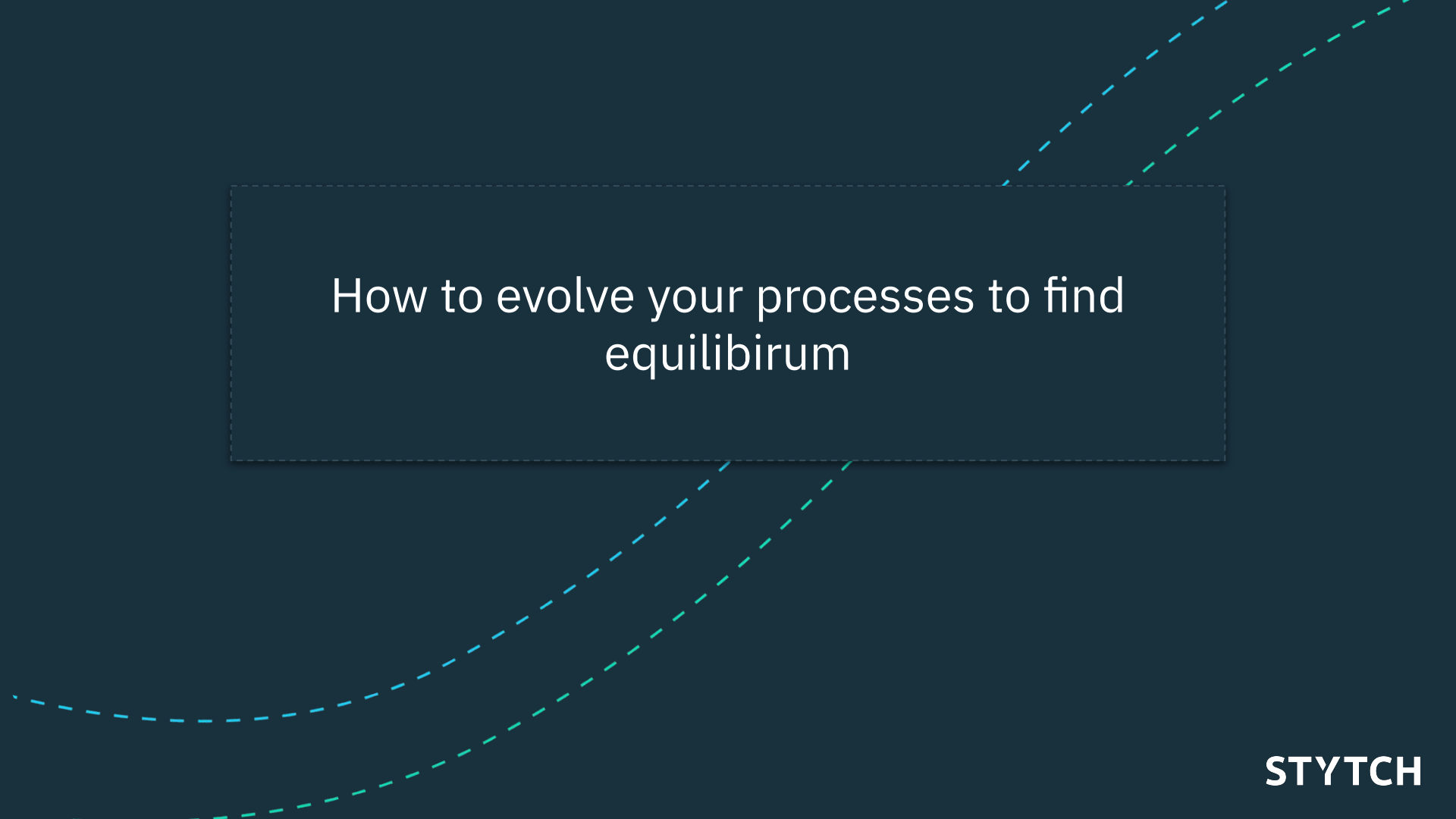
The team takes the spec phase very seriously, we go deep into figuring out what we're building and getting alignment. This saves us more time in the long run because weekly planning becomes much more straightforward.



We have a firehose communication culture. We try to create the minimum amount of overhead for cross team communication by standardizing what is shared and when so it's easy to keep track of high level progress and plans.



We emphasize product mindedness for our engineering team, as a dev tools company, engineers play a key role in defining our product. This upfront alignment empowers engineers to make decisions along the way.



How to evolve your processes to find
equilibrium

Finding equilibrium

What problems are you trying to solve? Dig in with your team to figure out what is causing friction today. Some examples of what that might look like:

1. Does too much time go to planning?
2. Does it take too long to get sign off on projects?
3. Do priorities change frequently?

1. How clear are your team's goals and priorities?

If you find yourself having to spend too much time planning week to week, more upfront alignment might be helpful.

2. Is there too much process that's getting in the way of progress?

There's a balance between upfront investment to ensure you're building the right thing, and having too many hoops to jump through.

3. Do you have enough process to ensure there's the right alignment on what's getting worked on and why?

More alignment on overall goals, for a team, project, or quarter can help to ensure you're not making constant u-turns.

How to land changes effectively



Don't experiment too much and change processes all the time, iteration and evolution are good, but chaos can ensue with too much change, too often



Get buy in from your team on the **goals** of the changes, frustration with new or more process can often stem from focusing on the short term implications, focusing on the impact of these changes over time and the resulting value can be effective



Develop processes with your team whenever possible, ownership over developing these processes can help to ensure they're 1) solving the right problems 2) there's buy in and adherence once they're rolled out

Thank you!



@juliannaelamb
stytech.com

STYTCH