# Tracing our journey into Distributed Tracing

•••

A tale of introducing new technologies in large scale organizations

Pedro Alves @ SRECon 2023

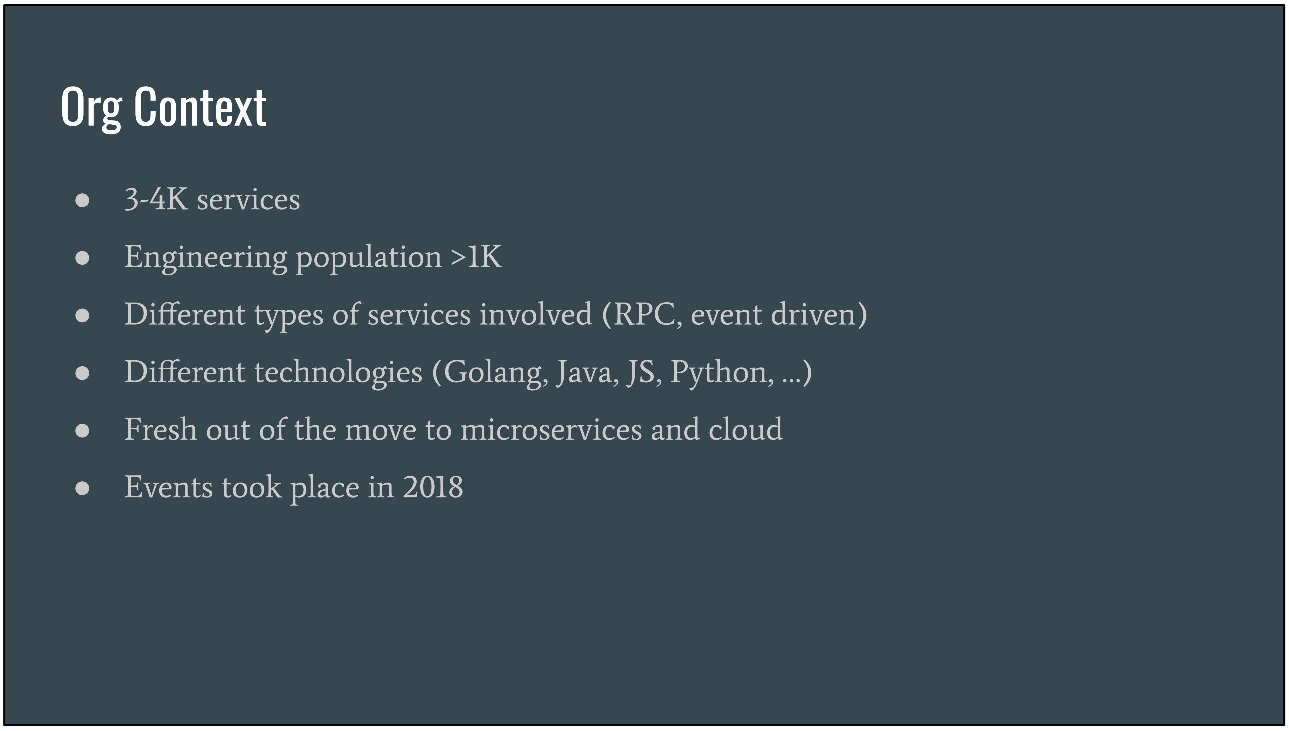
## Why am I here?

## “How did you adopt Distribute Tracing in your organization?”

In this talk I will walk you through the process of adopting Distributed Tracing in a large scale organization.

## “How did you adopt ${SOME_TECH} in your organization?”

Onboarding new technologies is something large companies do with some regularity. So we’ll use this story to identify key learnings and practices that could be reused to adopt other technologies in your own orgs.

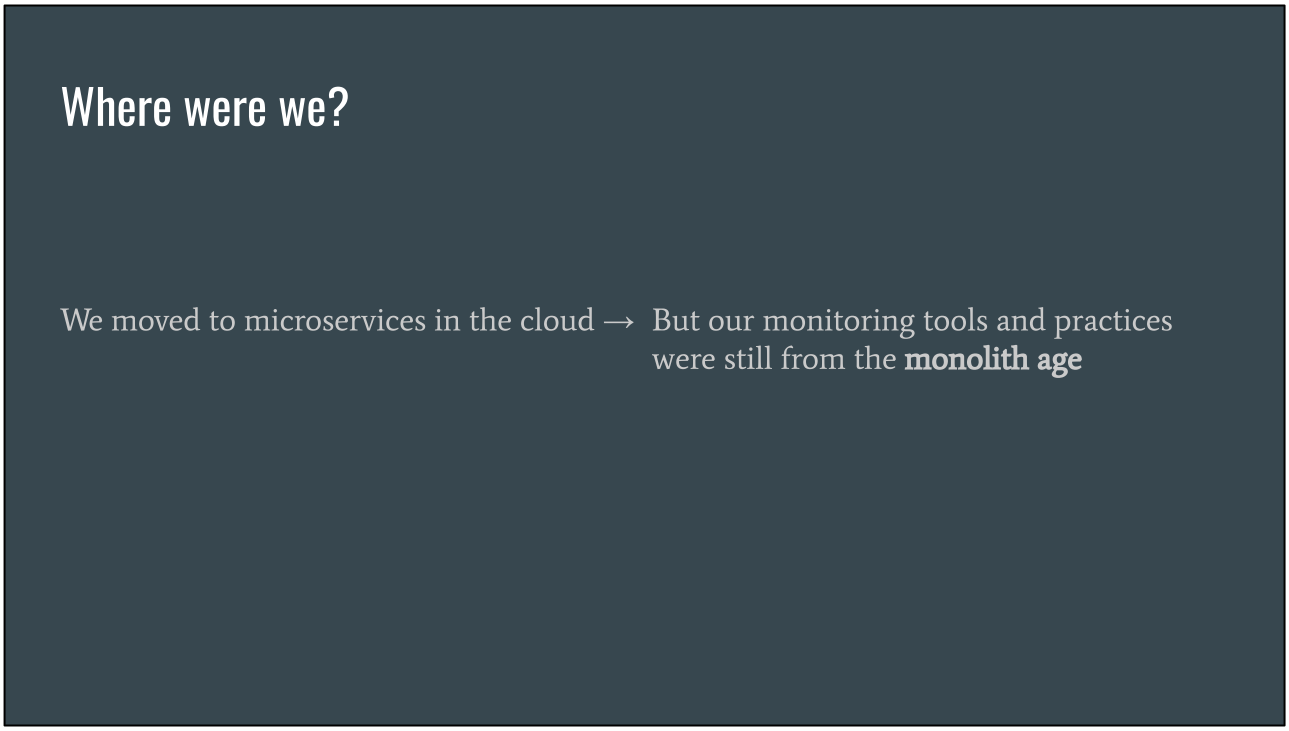


## Org Context

- 3-4K services
- Engineering population >1K
- Different types of services involved (RPC, event driven)
- Different technologies (Golang, Java, JS, Python, ...)
- Fresh out of the move to microservices and cloud
- Events took place in 2018

For some context, the organization where we rolled out distributed tracing was running 3-4K services, and had an engineering population of over 1K engineers. There were different types of services involved (RPC, event driven), as well as different technologies (Golang, Java, JS, Python, …)
Most of the events took place in 2018

“Do we have the right tools?”

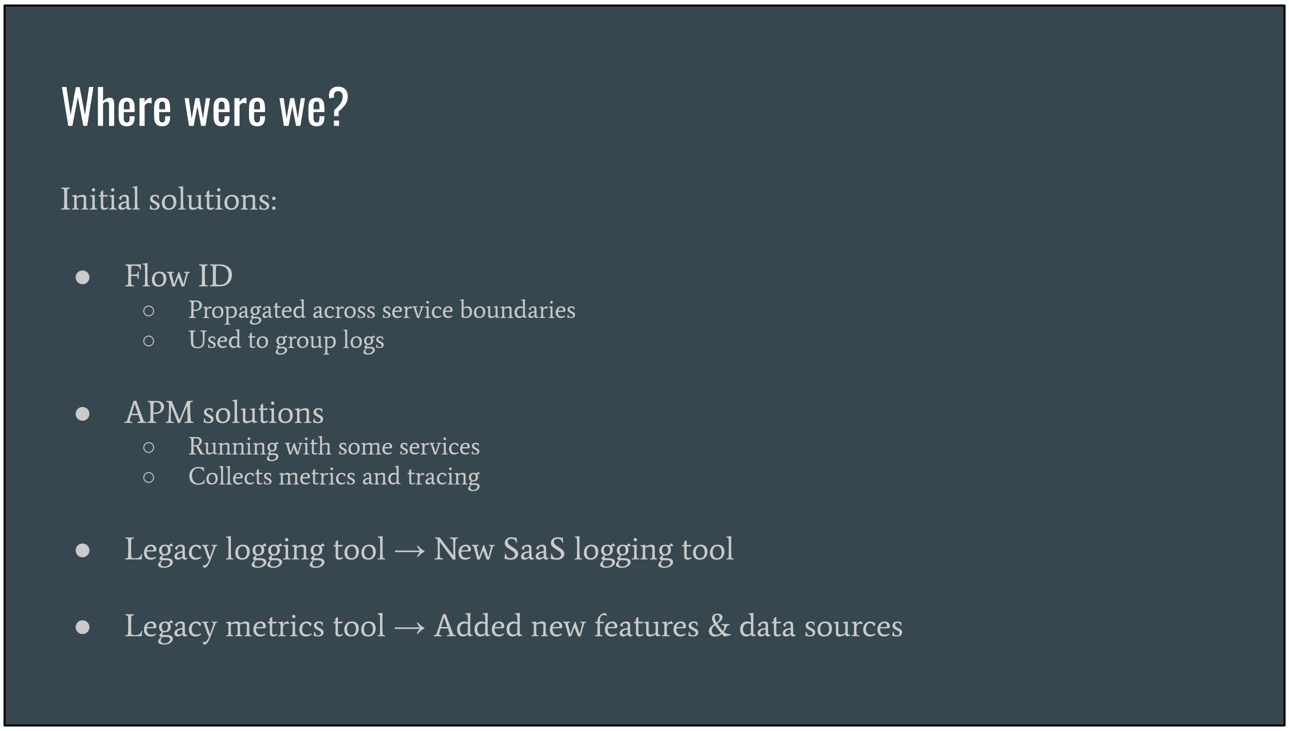


## Where were we?

We moved to microservices in the cloud → But our monitoring tools and practices were still from the **monolith age**

Back when the company migrated to Micro Services on the Cloud, with every team working in a “you build it you run it” fashion, all the previous tools we used to monitor our applications weren’t as helpful anymore.

The main challenge we faced at the time was adapting to a growing number of services, when all our monitoring processes and tooling were designed for a monolithic architecture.

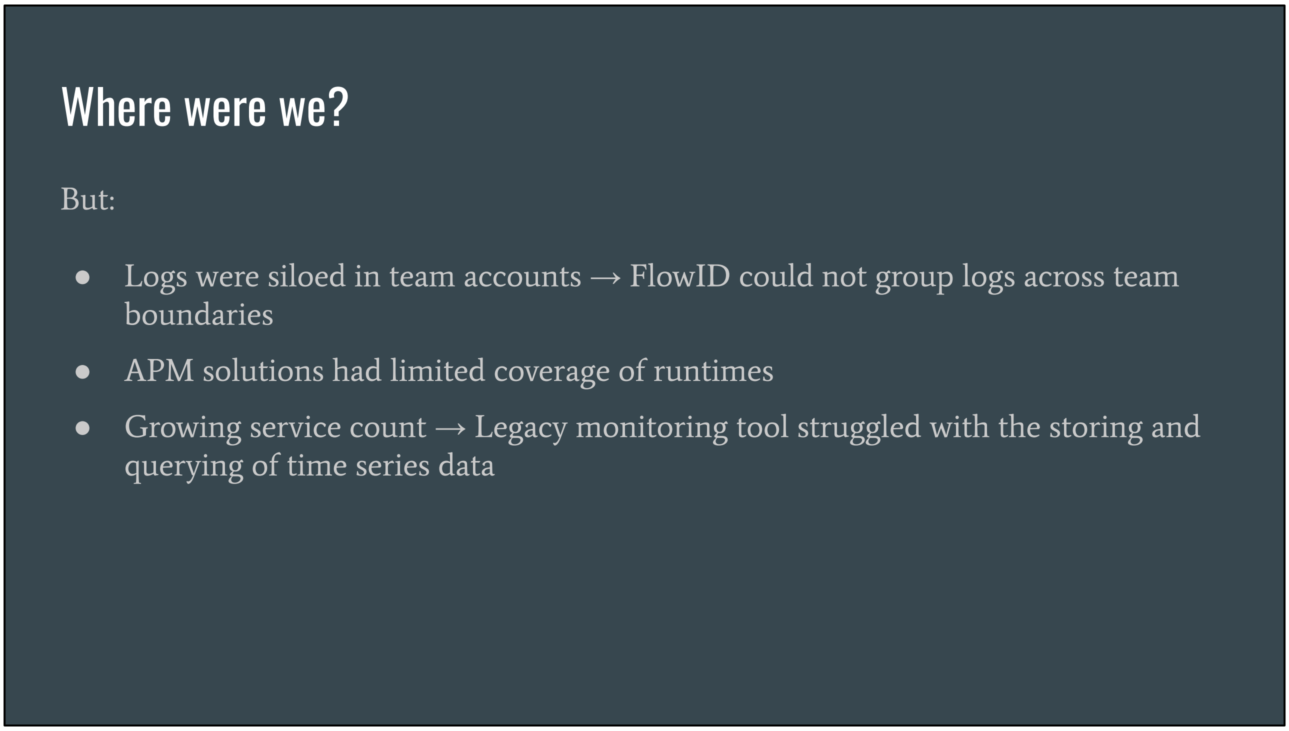


## Where were we?

Initial solutions:

- Flow ID
  - Propagated across service boundaries
  - Used to group logs
- APM solutions
  - Running with some services
  - Collects metrics and tracing
- Legacy logging tool → New SaaS logging tool
- Legacy metrics tool → Added new features & data sources

We did try different things

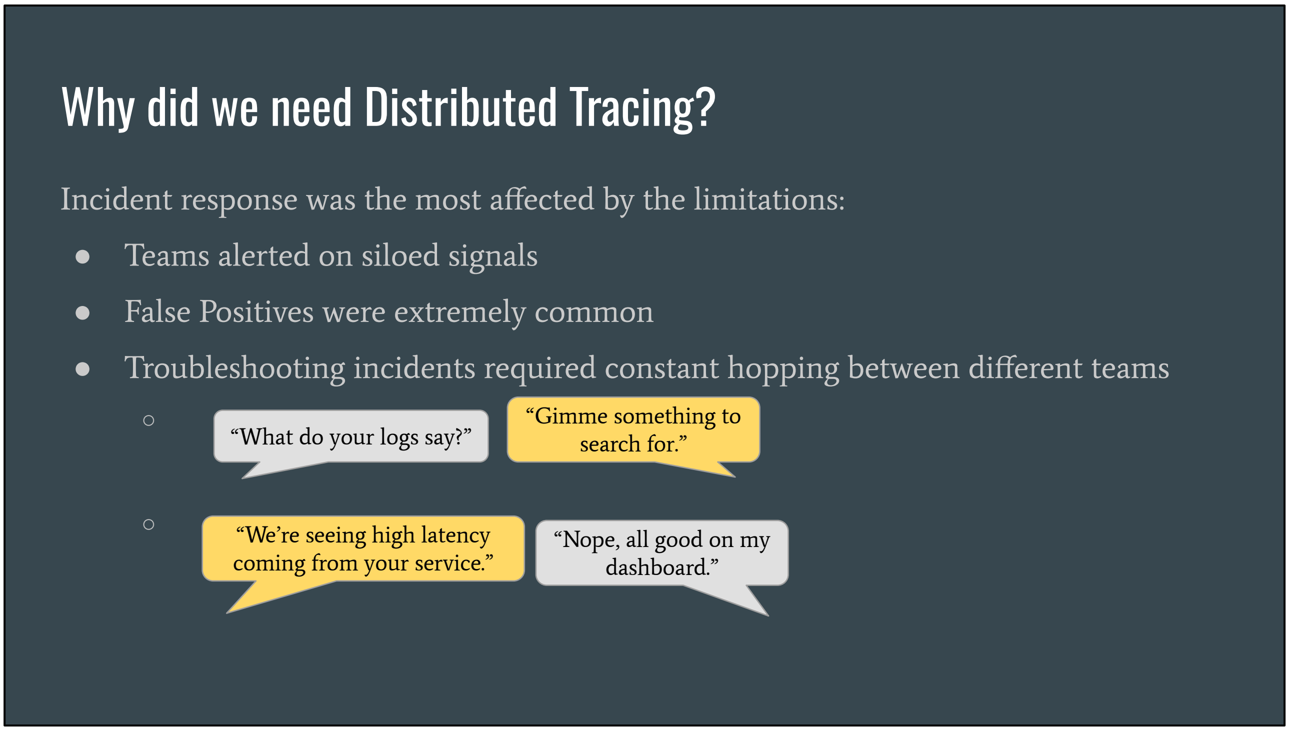


## Where were we?

But:

- Logs were siloed in team accounts → FlowID could not group logs across team boundaries
- APM solutions had limited coverage of runtimes
- Growing service count → Legacy monitoring tool struggled with the storing and querying of time series data

Whatever we did, we kept hitting limitations

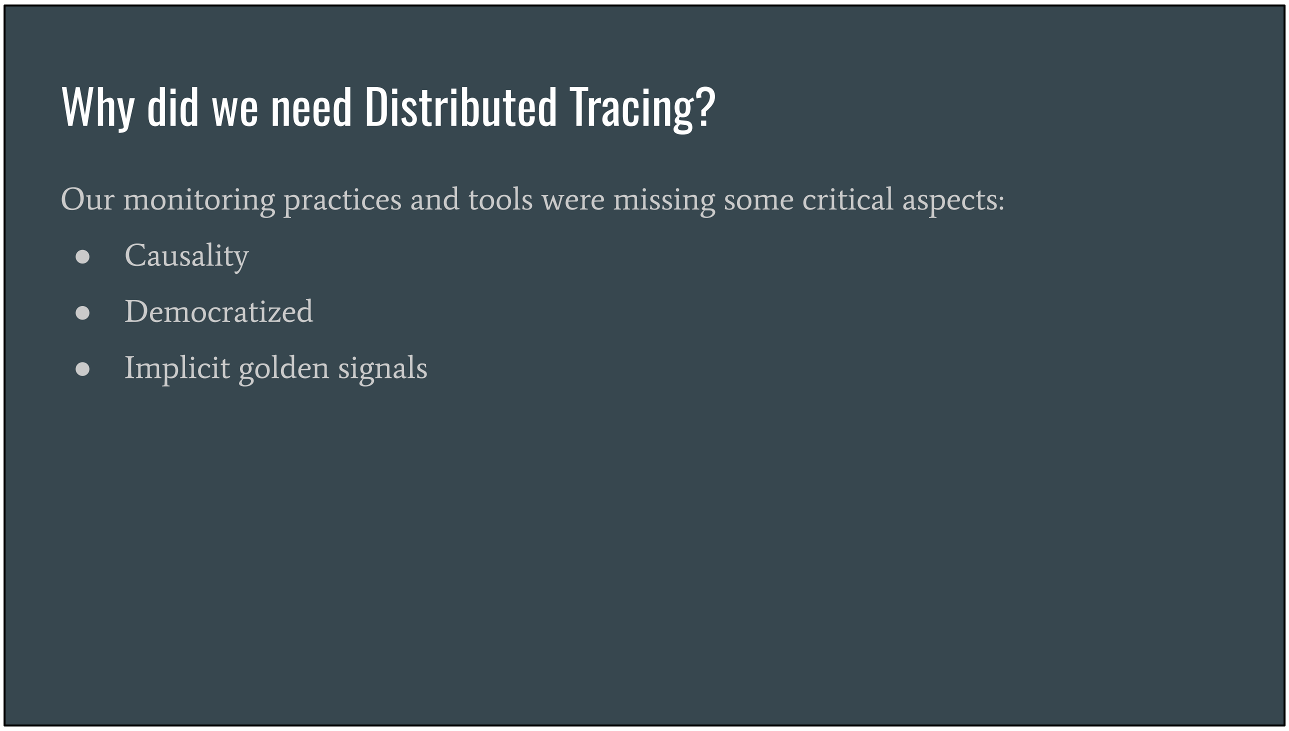


## Why did we need Distributed Tracing?

Incident response was the most affected by the limitations:

- Teams alerted on siloed signals
- False Positives were extremely common
- Troubleshooting incidents required constant hopping between different teams
  - “What do your logs say?” “Gimme something to search for.”
  - “We’re seeing high latency coming from your service.” “Nope, all good on my dashboard.”

The limitations we covered before effectively hindered reliability. Particularly with regards to incident response. Teams would alert on their siloed signals, with those signals often triggering False Positive Alerts. Understanding the source of a given incident in order to mitigate it, would regularly take longer than it should. Because teams could not understand each other’s signals, aligning on the source of the issue was more time consuming.



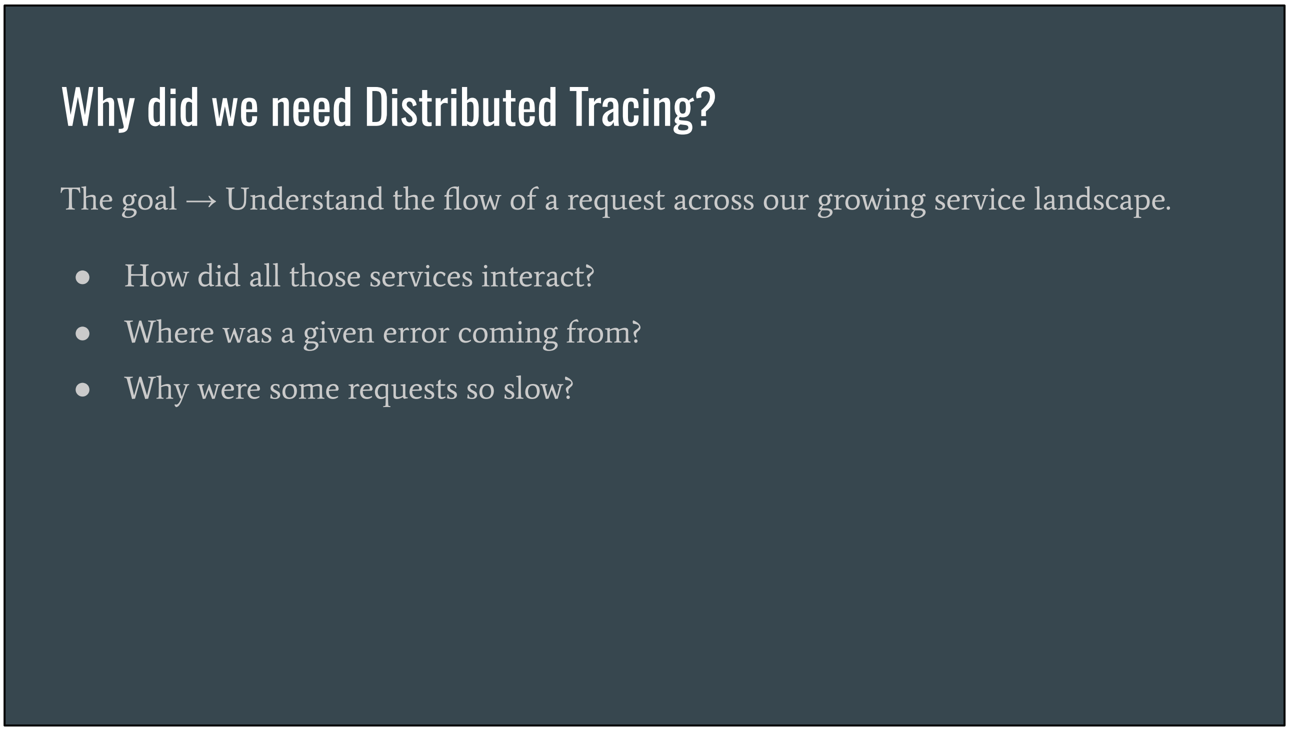
## Why did we need Distributed Tracing?

Our monitoring practices and tools were missing some critical aspects:

- Causality
- Democratized
- Implicit golden signals

Our monitoring practices and tools were missing some critical aspects:

- Causality - The telemetry we had was mostly isolated data points. Neither metrics nor logs would implicitly establish a link between one signal/event and the other.
- Democratized - Some logs were only available to some people. No one person could get a full picture from log data. Metrics data was available to everyone, but with different teams adopting different naming practices, and collecting and aggregating metrics differently, discovering metrics outside the ones emitted by your services was challenging.
- Implicit golden signals - Throughput, duration, errors, all required individual measurements. Nothing could provide even two of those signals implicitly.



## Why did we need Distributed Tracing?

The goal → Understand the flow of a request across our growing service landscape.

- How did all those services interact?
- Where was a given error coming from?
- Why were some requests so slow?

But, more to the point of this talk, our ultimate goal was understanding the flow of a request across that growing service landscape.

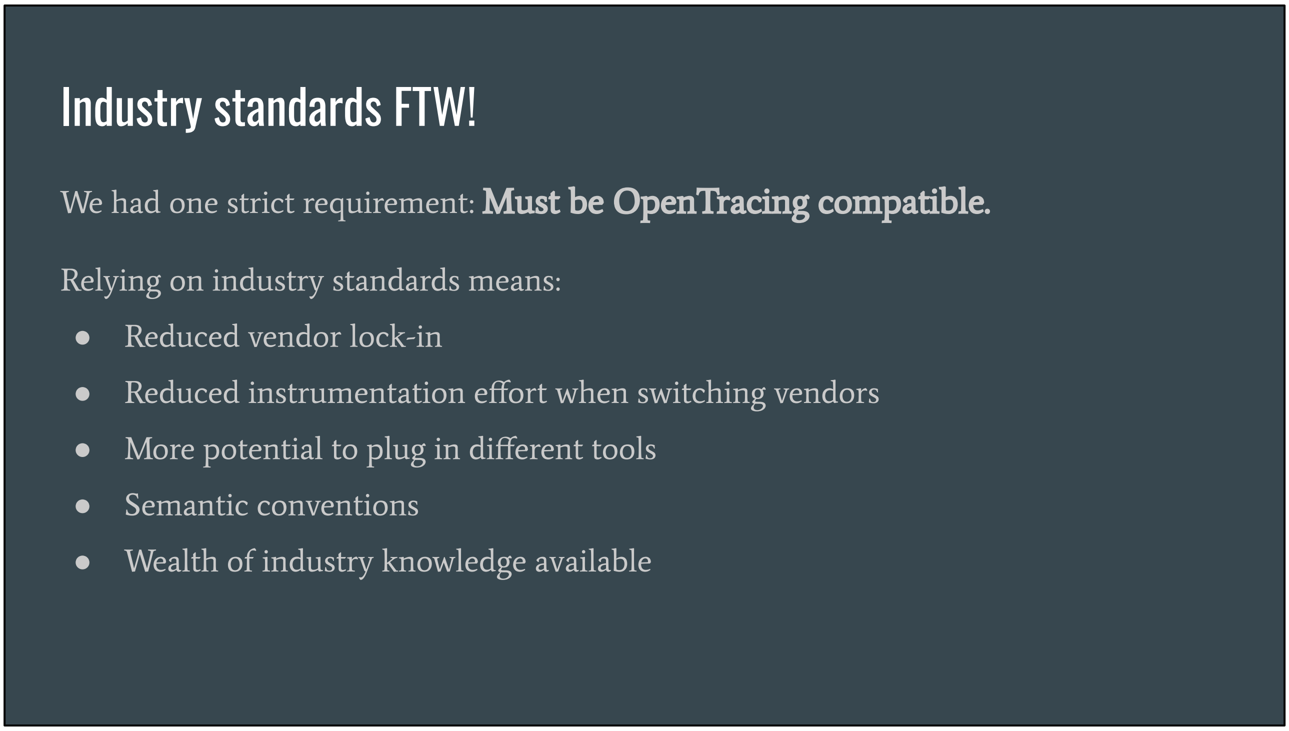
## Why did we need Distributed Tracing?

The goal → Understand the flow of a request across our growing service landscape.

- How did all those services interact?
- Where was a given error coming from?
- Why were some requests so slow?

We found the answer to these problems in Distributed Tracing

“Time for some experiments!”



## Industry standards FTW!

We had one strict requirement: **Must be OpenTracing compatible.**

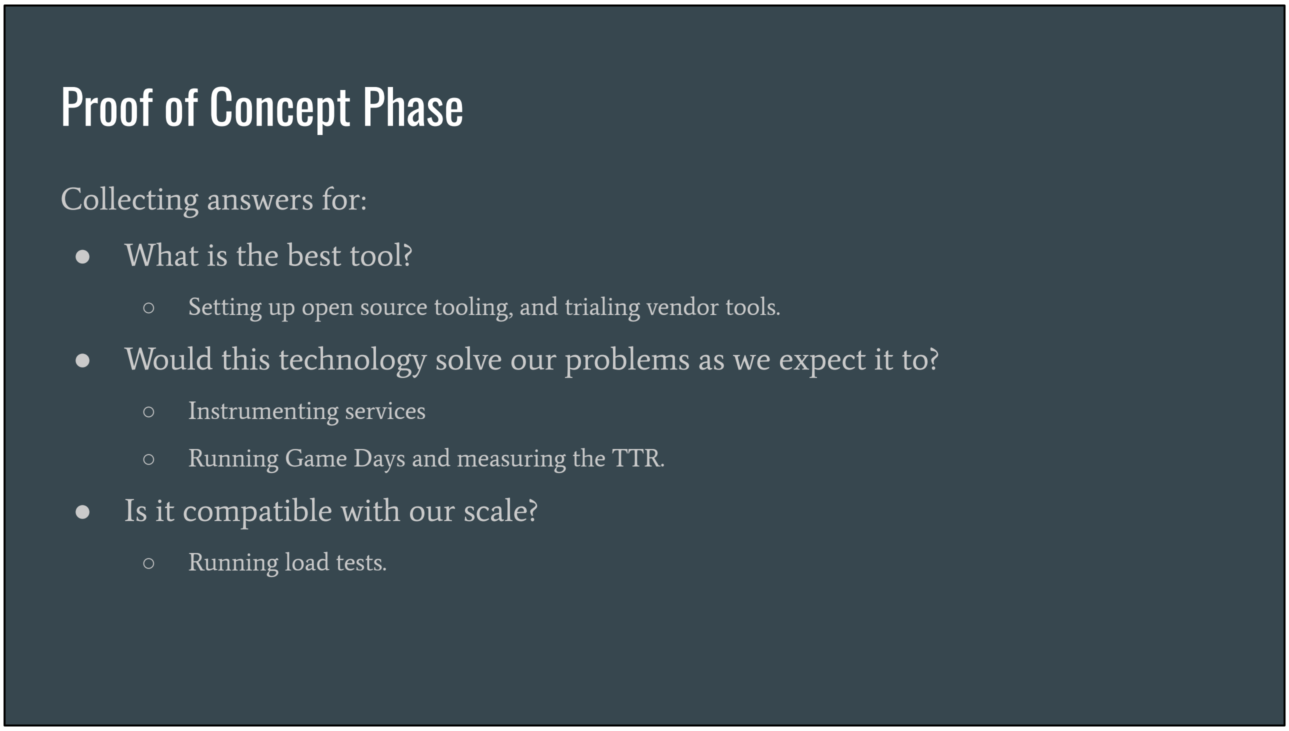
Relying on industry standards means:

- Reduced vendor lock-in
- Reduced instrumentation effort when switching vendors
- More potential to plug in different tools
- Semantic conventions
- Wealth of industry knowledge available

There was one thing in our requirement list for whatever Distributed Tracing solution we went for: it had to be OpenTracing compatible. The reason was strategic.

After going through a few monitoring vendors in recent years, we learned that reinstrumenting services was a considerable cost. We wanted to avoid the same cost in case we needed to migrate to another solution. Adhering to the standard would make that a possibility. We could, with minimal changes to application setup (tracer change), change to a new vendor, without having to change any of the instrumentation of the application code itself.
Admittedly, at our service count at the time (~3-4K services), this would still take a long time to redeploy the entire fleet. Still, the effort was much reduced, compared to instrument services all over again.
Today, with OpenTelemetry and its collectors, things could even be simpler to migrate.

There are more benefits in using an industry standard. The standardized data model allows for easier integration with other tools.
The semantic conventions guide with instrumentation, and define how to represent common concepts.
Plus, there is a wealth of industry knowledge.



## Proof of Concept Phase

Collecting answers for:

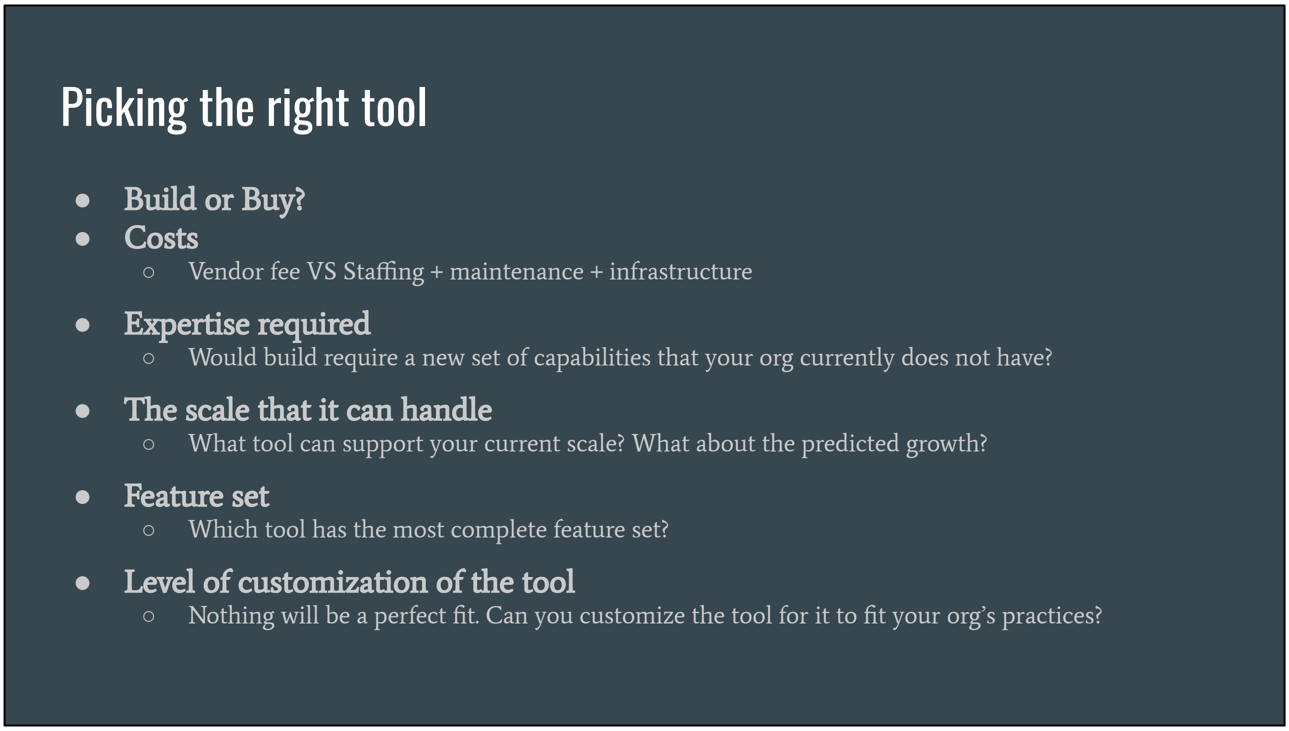
- What is the best tool?
  - Setting up open source tooling, and trialing vendor tools.
- Would this technology solve our problems as we expect it to?
  - Instrumenting services
  - Running Game Days and measuring the TTR.
- Is it compatible with our scale?
  - Running load tests.

Experiments were ran during a Proof of Concept phase.

We ran a few experiments with open source tooling. We tried to understand what would it take to run the whole thing ourselves, at the scale we were at the time, and beyond. In parallel, we also reached out to vendors and tried their solutions as well.

To try and demonstrate the usefulness of Tracing, we ran a couple of game days measuring the TTR for each exercise. This KPI turned out to be a poor choice, as the learnings were not useful. Nevertheless, the visibility Tracing provided was in line with what we were looking to get, which was enough to maintain the motivation in the project.

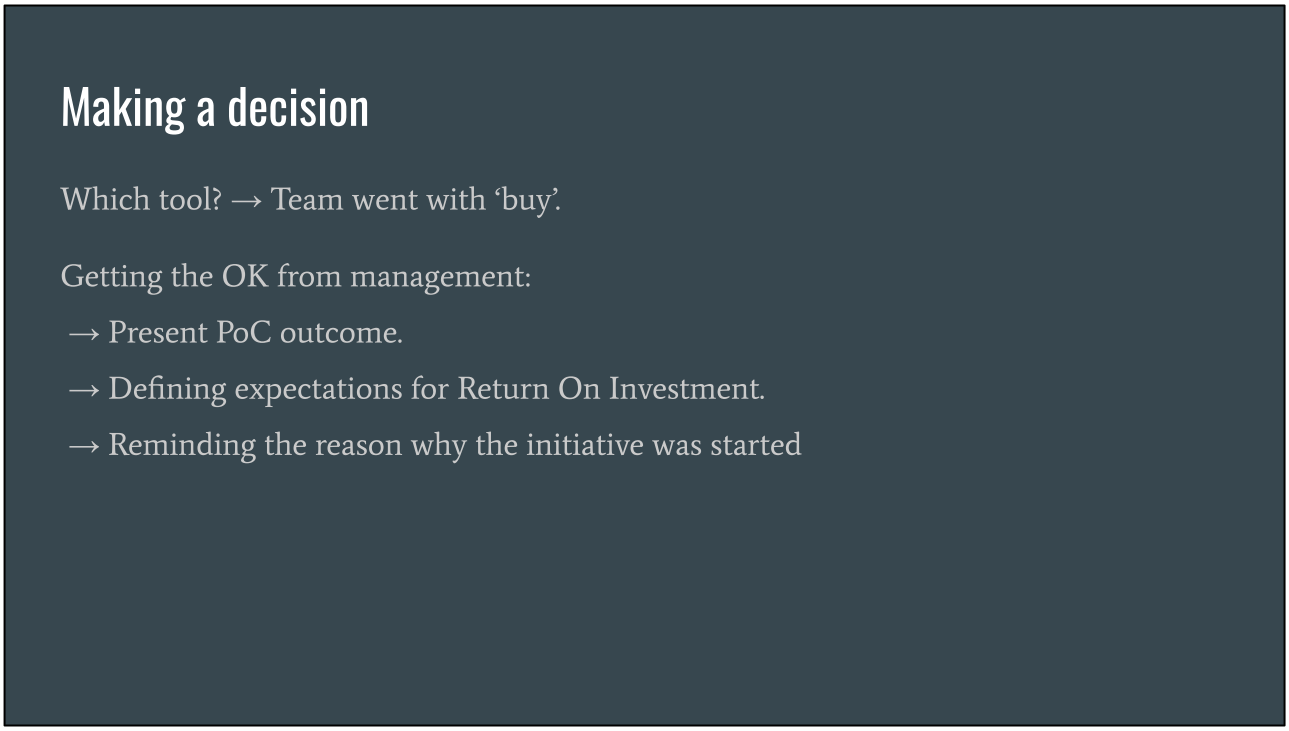
We also ran load tests to understand if it would be compatible with our scale, and our predicted growth.



## Picking the right tool

- **Build or Buy?**
- **Costs**
  - Vendor fee VS Staffing + maintenance + infrastructure
- **Expertise required**
  - Would build require a new set of capabilities that your org currently does not have?
- **The scale that it can handle**
  - What tool can support your current scale? What about the predicted growth?
- **Feature set**
  - Which tool has the most complete feature set?
- **Level of customization of the tool**
  - Nothing will be a perfect fit. Can you customize the tool for it to fit your org's practices?

Tracing data needs to be ingested, processed, and visualised. To do all of that, we need the right tool.
The tools we experimented with, open source or third party, were evaluated on the same criteria.



## Making a decision

Which tool? → Team went with ‘buy’.

Getting the OK from management:

→ Present PoC outcome.

→ Defining expectations for Return On Investment.

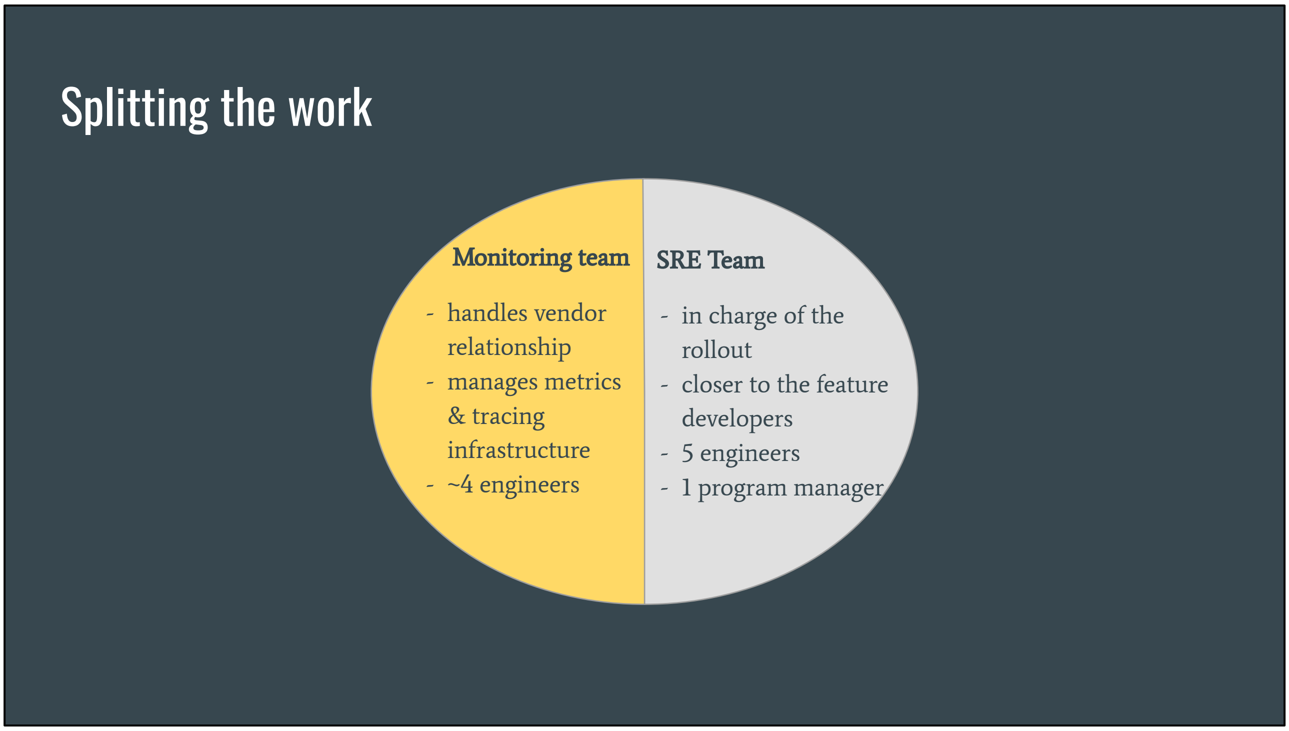
→ Reminding the reason why the initiative was started

As for Build or Buy, the team decided in favor of ‘buying’.

When we collected all the numbers we could, and documented our recommendations, it was time to bring it up to management so that a decision could be made.

We presented the PoC outcome, our learnings and recommendations.
We also gave some idea of the return on investment in tracing. This was mostly reminding management of why we were looking at tracing in the first place.
We highlighted the limitations teams were facing.
How on-call work is that much more difficult.
How reliability was affected.
And, of course, how Distributed Tracing could help us with that.

“Let the rollout commence!”



## Splitting the work

**Monitoring team**

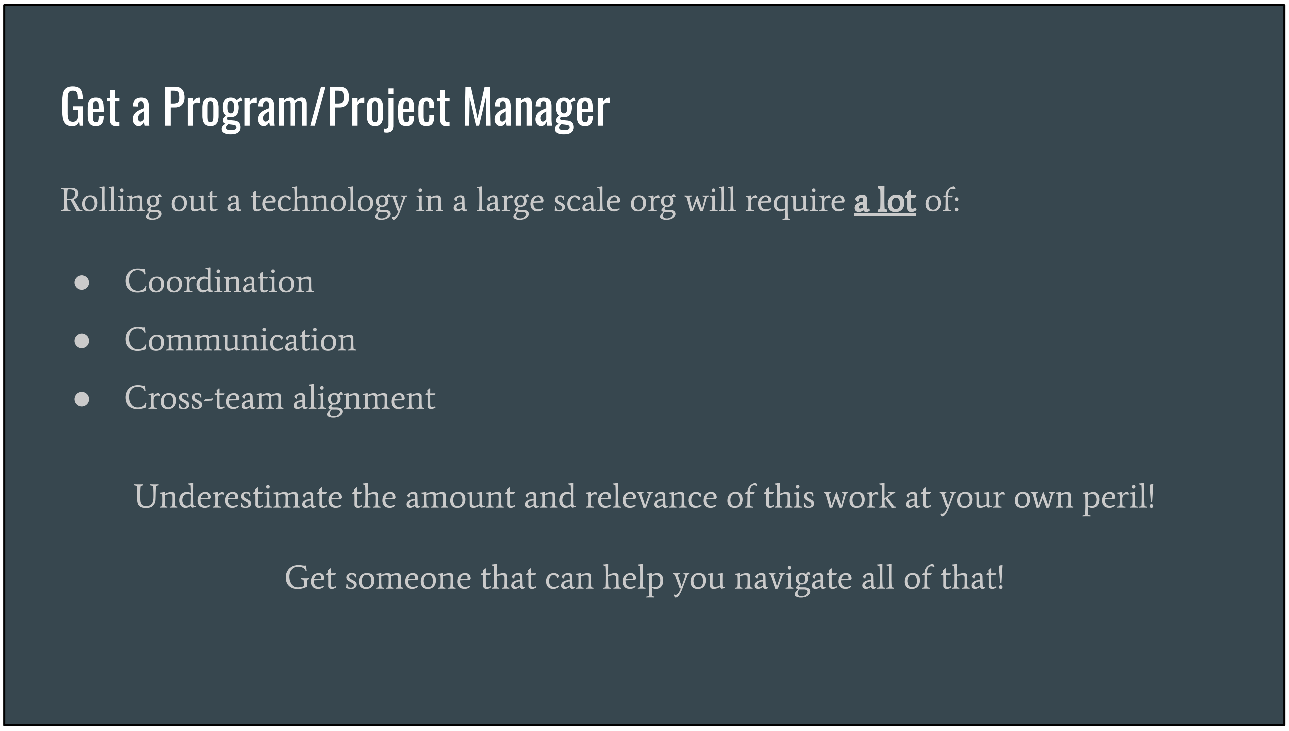
- handles vendor relationship
- manages metrics & tracing infrastructure
- ~4 engineers

**SRE Team**

- in charge of the rollout
- closer to the feature developers
- 5 engineers
- 1 program manager

The work described so far was driven by the team managing all of the monitoring tools, and infrastructure. Right around the time the PoC was about to wrap up, an SRE team had just been created. After getting the go-ahead from management, the SRE team was tasked with the rollout of Distributed Tracing to the rest of the engineering org - that is actually when I came into the project. This assignment did not come out of the blue. Members of the new SRE team were already working together with the team owning the monitoring tools, so there was a bit of history already.

This may seem a bit of a detail, but there was an important ingredient that this SRE team brought: we were much closer to the feature developers (the ‘customers’), than the infrastructure teams. This proximity gave us an extra insight on how to bring this new technology to the wider engineering population.



## Get a Program/Project Manager

Rolling out a technology in a large scale org will require **a lot** of:

- Coordination
- Communication
- Cross-team alignment

Underestimate the amount and relevance of this work at your own peril!

Get someone that can help you navigate all of that!

Speaking of the SRE team, we had 5 engineers, and one Program Manager. For such a large scale endeavor, having a Program (or Project) Manager can really make a difference. In our case it certainly did. The amount of coordination, communication, and cross-team alignment that is required is not to be underestimated. Get someone that can help you navigate all of that.

## Finding Leverage

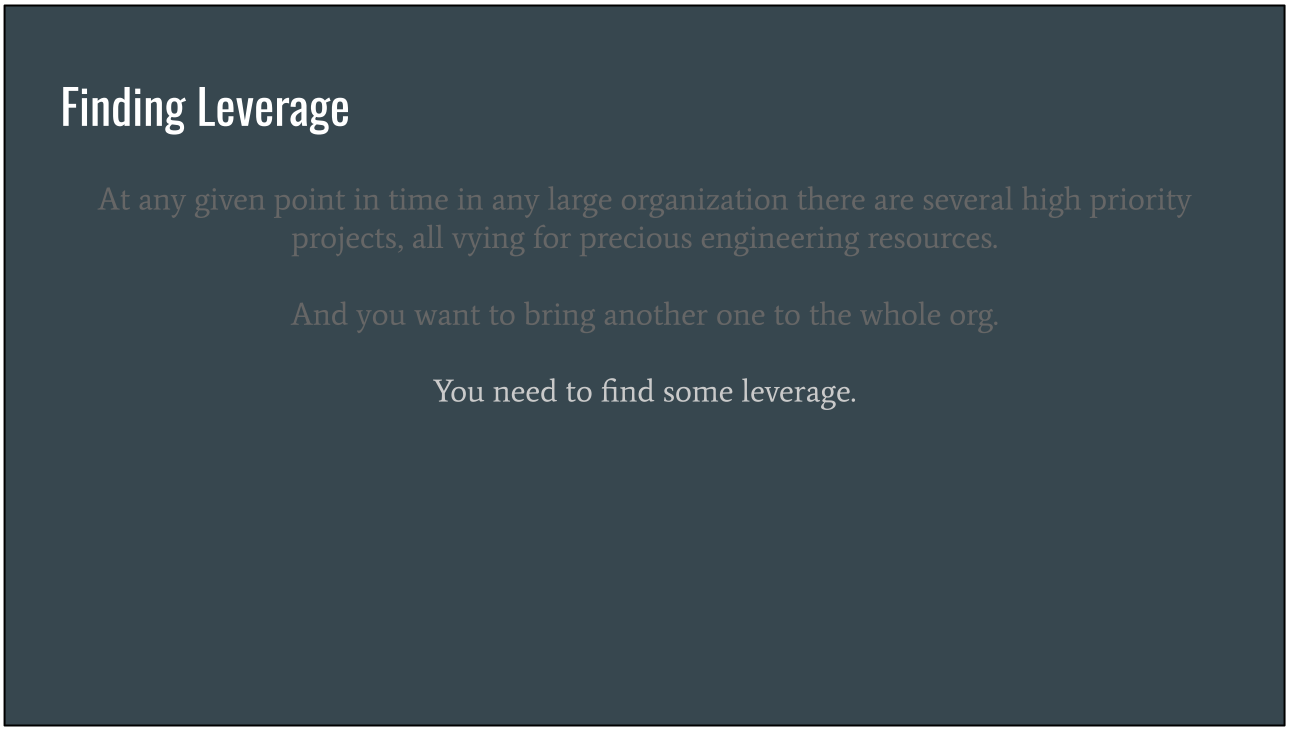
At any given point in time in any large organization there are several high priority projects, all vying for precious engineering resources.

## Finding Leverage

At any given point in time in any large organization there are several high priority projects, all vying for precious engineering resources.

And you want to bring another one to the whole org.

When you are rolling out a technology for an entire company, and you need support from all those teams, it's important to remember that every one of those teams will have 1 to 3 "Number 1" priorities. How will you get support for your rollout, when you're competing with all those projects?



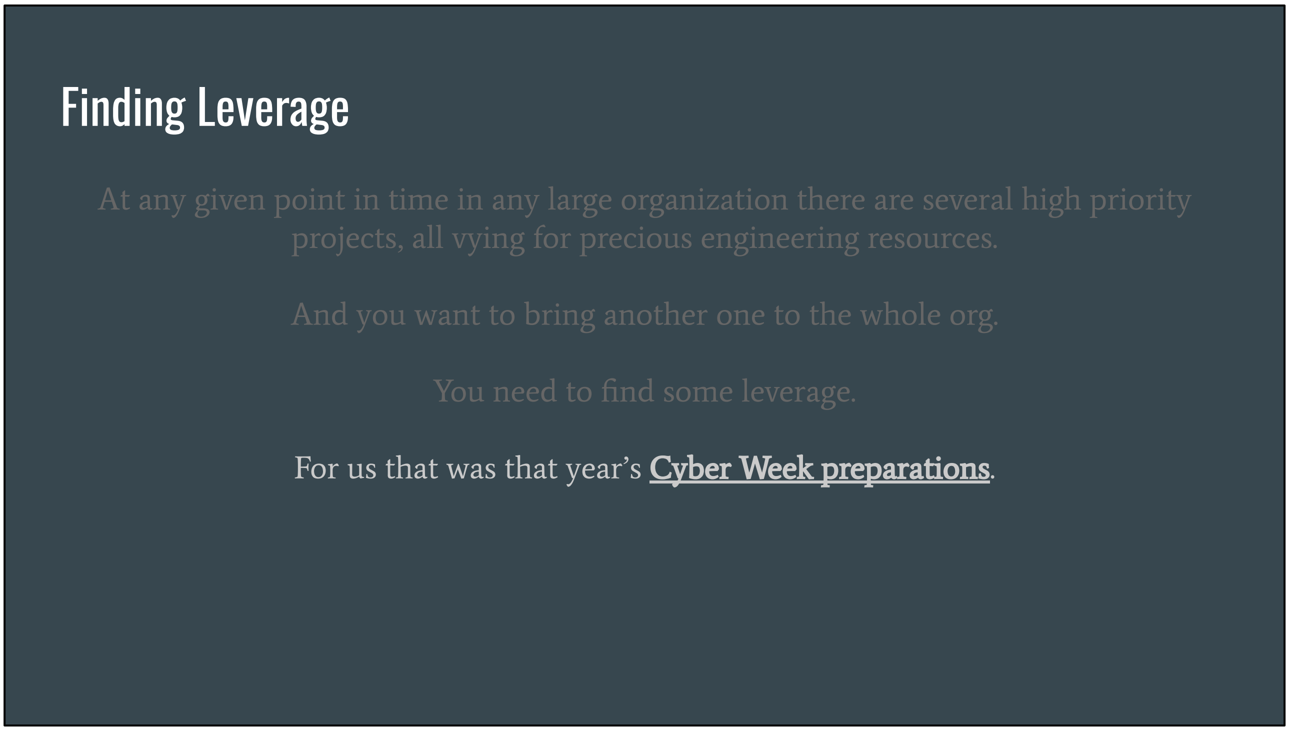
## Finding Leverage

At any given point in time in any large organization there are several high priority projects, all vying for precious engineering resources.

And you want to bring another one to the whole org.

You need to find some leverage.

We needed to get everyone’s attention, and find our way to the top of their list of priorities.



## Finding Leverage

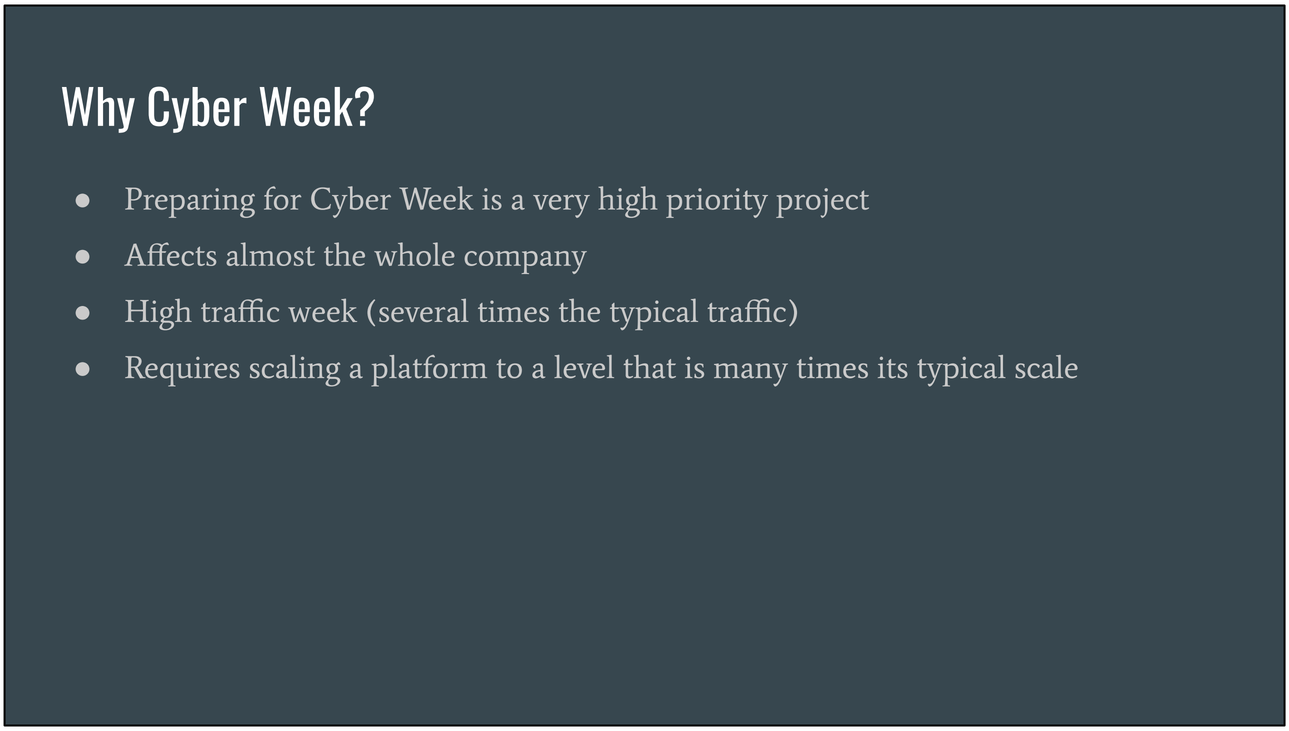
At any given point in time in any large organization there are several high priority projects, all vying for precious engineering resources.

And you want to bring another one to the whole org.

You need to find some leverage.

For us that was that year's **Cyber Week preparations**.

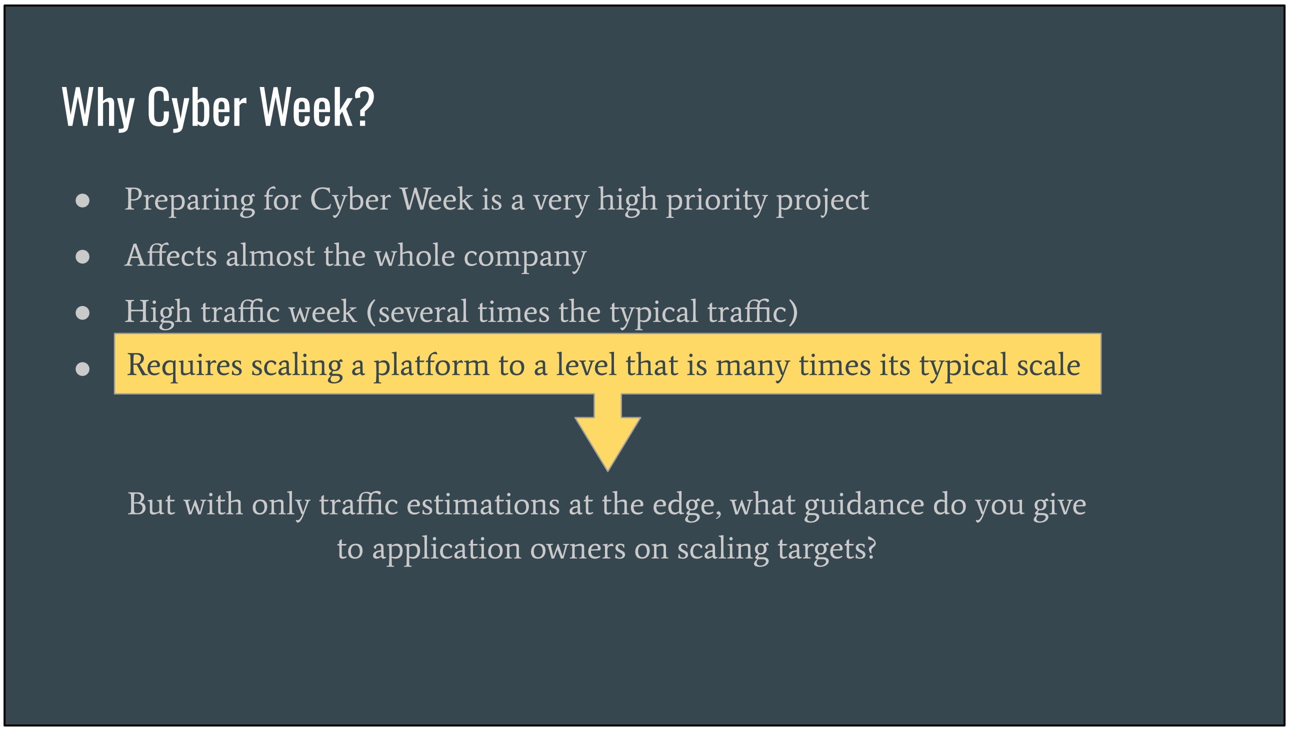
Fortunately for us, one of the projects that was very high in the priority list for every team, was something that we could piggy back on. It was the preparation for that year's Cyber Week.



## Why Cyber Week?

- Preparing for Cyber Week is a very high priority project
- Affects almost the whole company
- High traffic week (several times the typical traffic)
- Requires scaling a platform to a level that is many times its typical scale

Cyber Week is a week of high traffic on e-commerce systems that have these campaigns. The traffic can be several times the traffic on even a busy day in any other time of the year. The impact of an incident during that week will have a proportionally higher impact on revenue and reputation.

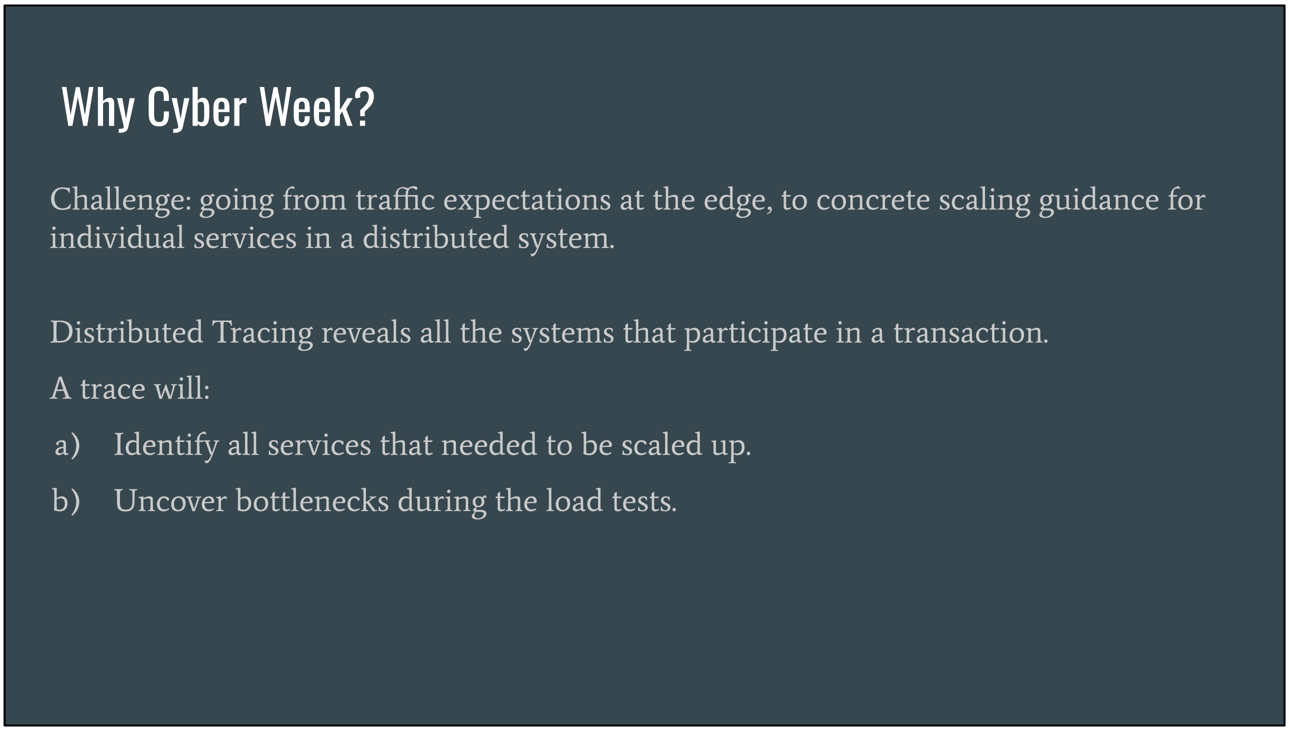


## Why Cyber Week?

- Preparing for Cyber Week is a very high priority project
- Affects almost the whole company
- High traffic week (several times the typical traffic)
- Requires scaling a platform to a level that is many times its typical scale

But with only traffic estimations at the edge, what guidance do you give to application owners on scaling targets?

To prepare for Cyber Week, and scale our platform we only had rough estimates of edge traffic. We still had to find the right target for scaling the different services


## Why Cyber Week?

Challenge: going from traffic expectations at the edge, to concrete scaling guidance for individual services in a distributed system.

Distributed Tracing reveals all the systems that participate in a transaction.

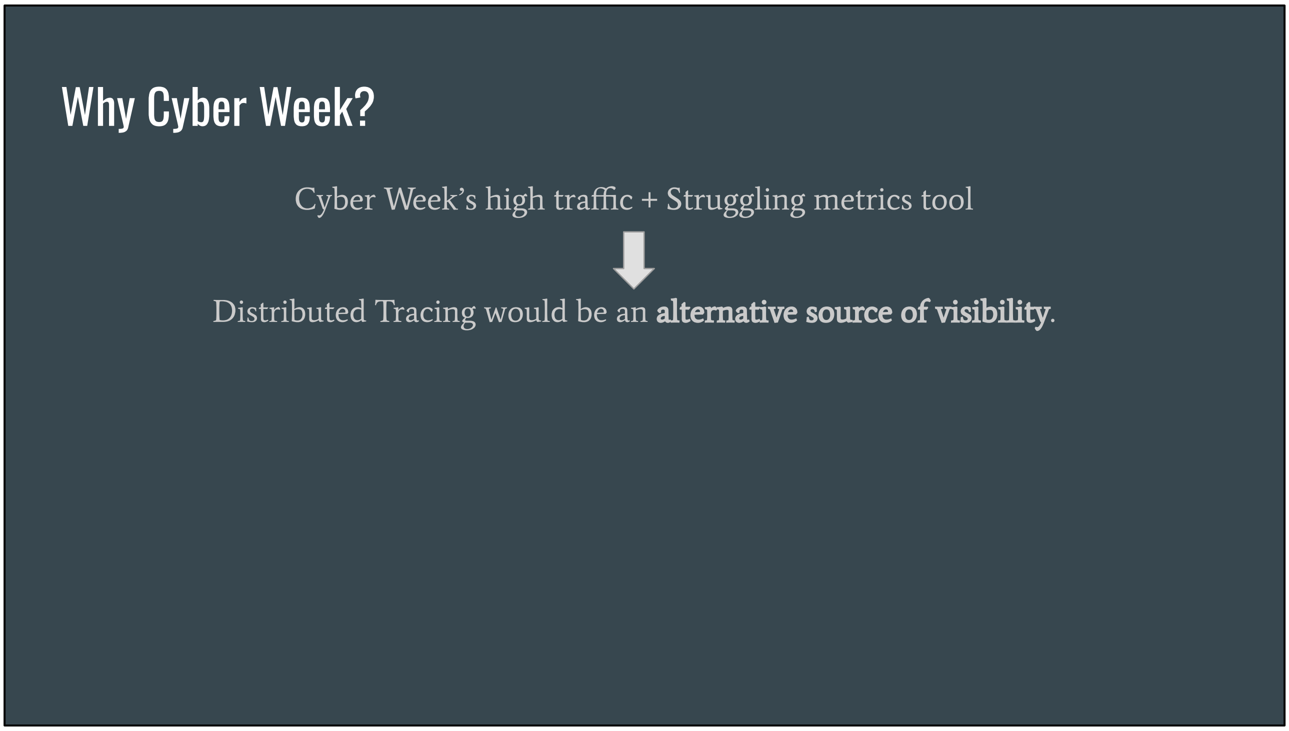
A trace will:

a) Identify all services that needed to be scaled up.
b) Uncover bottlenecks during the load tests.

To ensure the platform will handle the expected traffic we need to scale it to a given level, and validate that scale against the expected throughput with load tests.

We argued that Tracing could help with the scaling efforts by a) identifying the services that need to be scaled, and b) identifying bottlenecks during load tests.

The help with the load tests was part of the reason why management agreed on moving forward with this



## Why Cyber Week?

Cyber Week's high traffic + Struggling metrics tool

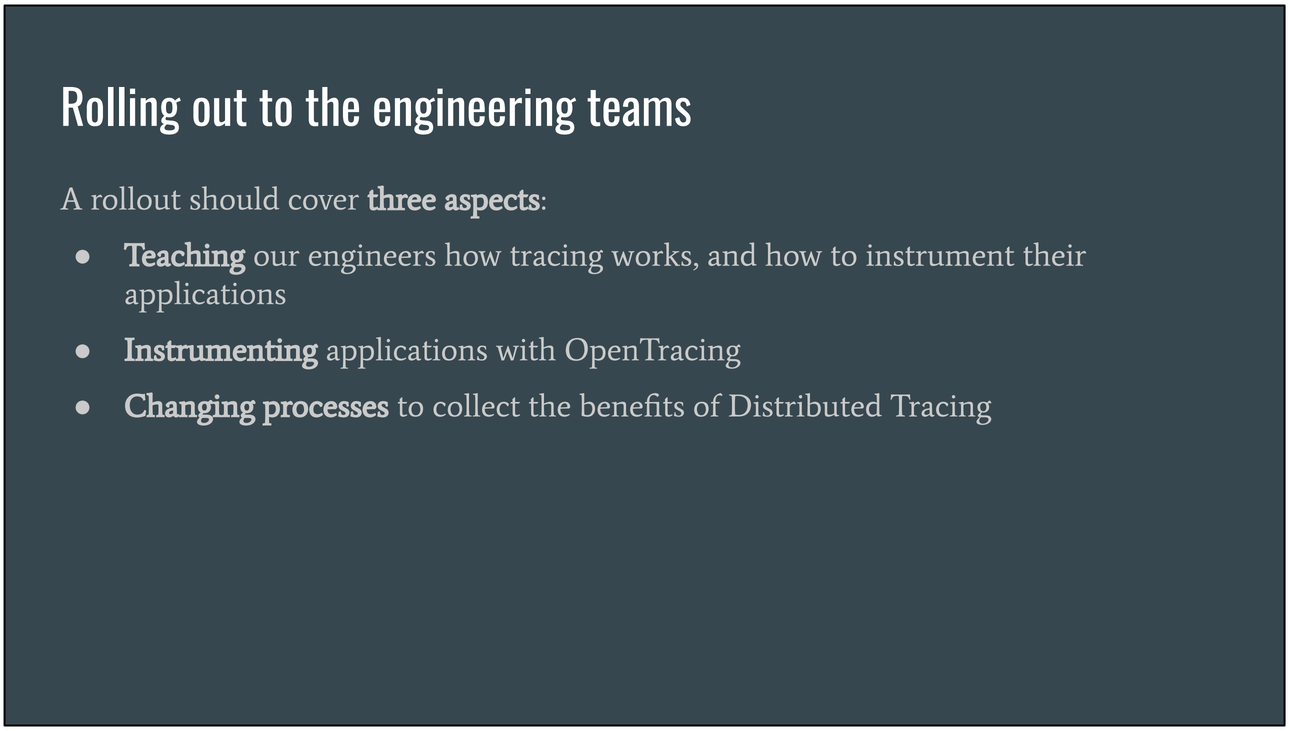
⬇

Distributed Tracing would be an **alternative source of visibility**.

Given the struggles that our legacy monitoring system had with even the typical scale, there were concerns that scaling all those services would overload our monitoring system. Of course, Distributed Tracing would not replace that. But it would certainly bring an alternative, so we wouldn't be blind if/when the monitoring system would have an outage.

The usefulness of Tracing for the Cyber Week preparation efforts was enough to make rolling out Distributed Tracing to the affected systems a priority. We had found our leverage.

“Meeting the teams.”

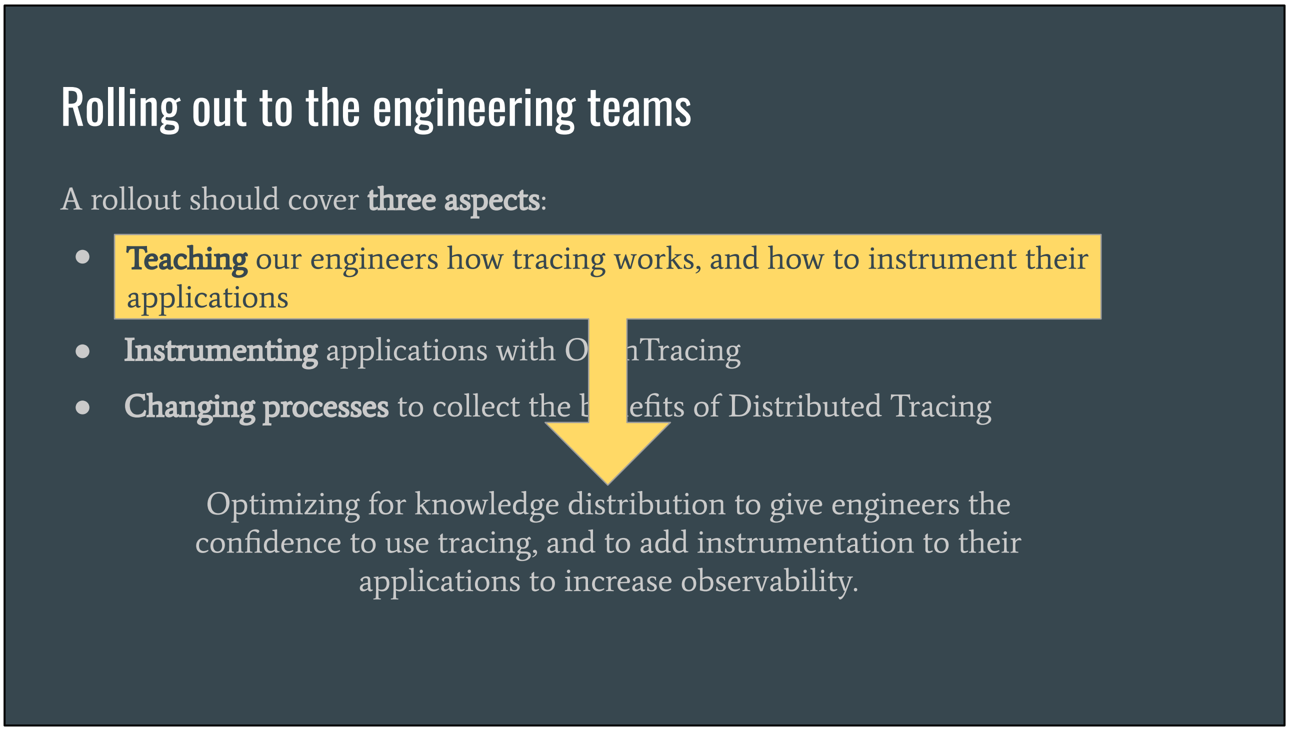


## Rolling out to the engineering teams

A rollout should cover **three aspects**:

- **Teaching** our engineers how tracing works, and how to instrument their applications
- **Instrumenting** applications with OpenTracing
- **Changing processes** to collect the benefits of Distributed Tracing

We brainstormed a bit how to approach this rollout. We soon realized that **a proper rollout had to cover three aspects:**



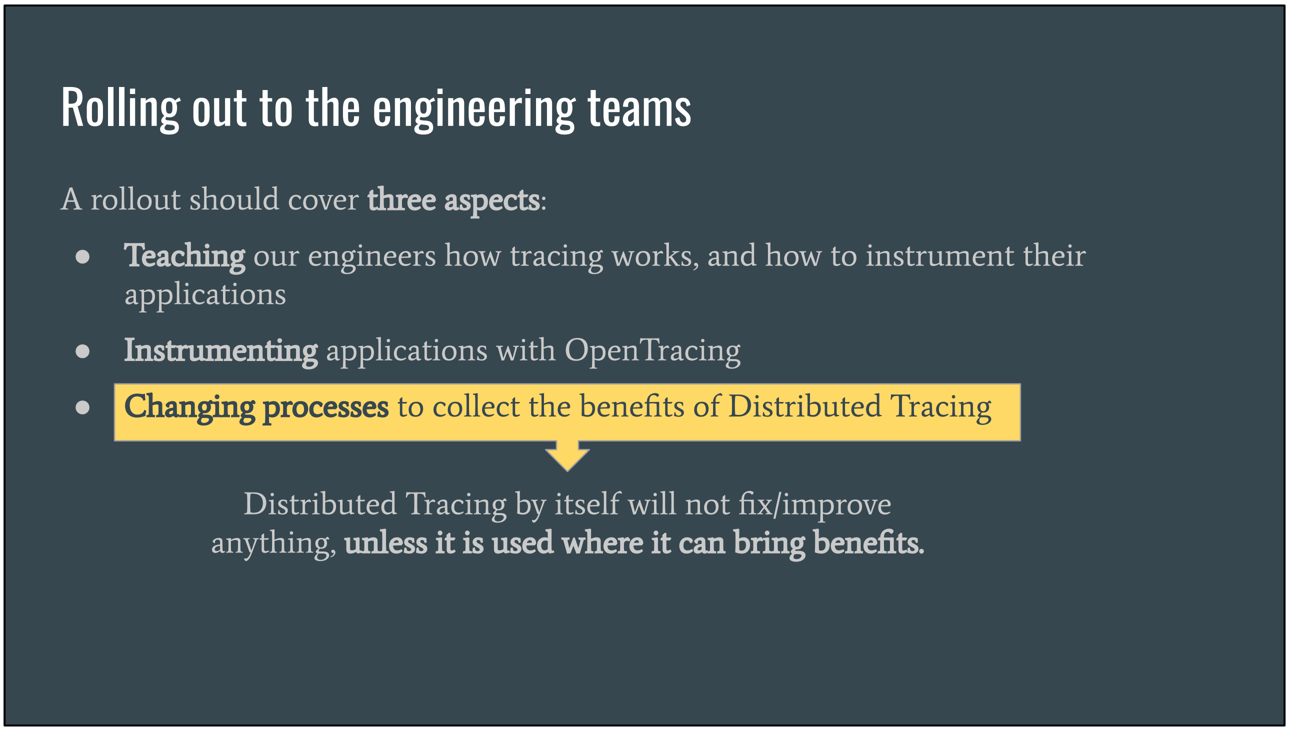
## Rolling out to the engineering teams

A rollout should cover **three aspects**:

- **Teaching** our engineers how tracing works, and how to instrument their applications
- **Instrumenting** applications with OpenTracing
- **Changing processes** to collect the benefits of Distributed Tracing

Optimizing for knowledge distribution to give engineers the confidence to use tracing, and to add instrumentation to their applications to increase observability.

This may sound slow - and it is, in the short term -, but we optimized for knowledge distribution. Our guiding principle was to grant engineers with enough knowledge so that tracing would not be some black box that they had no idea of how it worked. **We wanted them to use tracing, and we wanted them to feel comfortable adding more instrumentation to their applications to increase observability.**



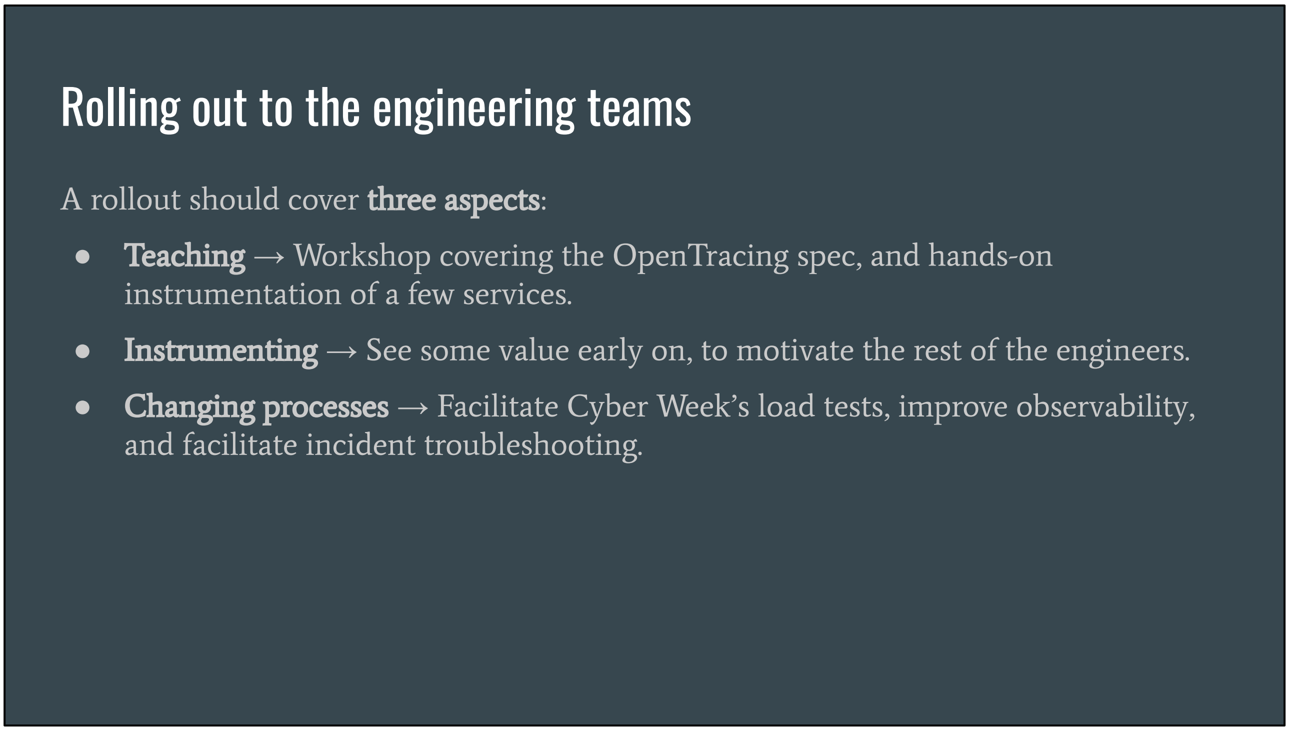
## Rolling out to the engineering teams

A rollout should cover **three aspects**:

- **Teaching** our engineers how tracing works, and how to instrument their applications
- **Instrumenting** applications with OpenTracing
- **Changing processes** to collect the benefits of Distributed Tracing

Distributed Tracing by itself will not fix/improve anything, **unless it is used where it can bring benefits.**

This last point is key. Distributed Tracing by itself will not fix/improve anything, unless it is used where it can bring benefits.



## Rolling out to the engineering teams

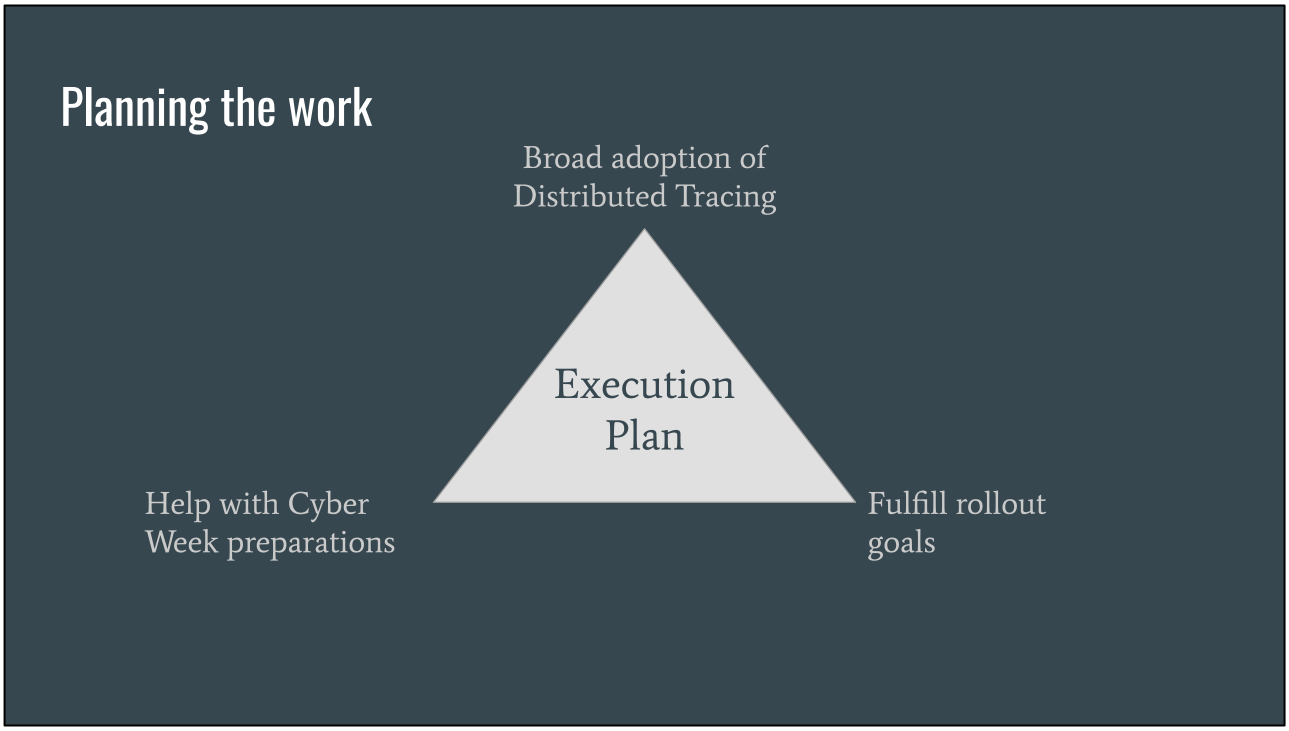
A rollout should cover **three aspects**:

- **Teaching** → Workshop covering the OpenTracing spec, and hands-on instrumentation of a few services.
- **Instrumenting** → See some value early on, to motivate the rest of the engineers.
- **Changing processes** → Facilitate Cyber Week's load tests, improve observability, and facilitate incident troubleshooting.

Our approach to start spreading the knowledge about OpenTracing across the org was a workshop. In it, we covered the basics of OpenTracing, coupled with a hands-on part where we instrumented a few services (very similar to the content we brought to SRECon in 2019).

With instrumenting applications we were aiming to see some value early on, to motivate the rest of the org. We could not instrument everything ourselves, so we relied on generating enough interest in Tracing to get enough people to instrument their applications.

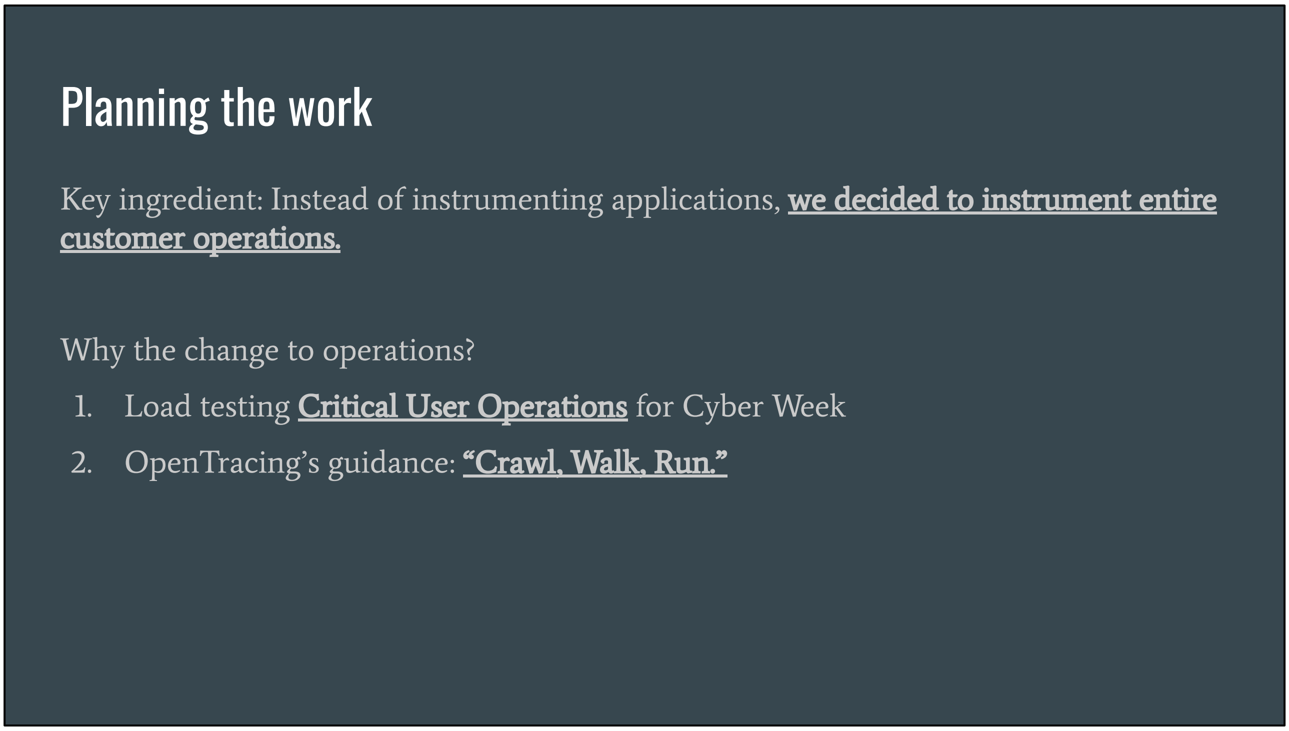
Finally, regarding the processes where we'd use Tracing to collect benefits, that was easy. That is where we started the whole thing. We wanted to increase observability, and facilitate incident troubleshooting. On top of that we also had our promised contribution to the Cyber Week project.



## Planning the work

Broad adoption of Distributed Tracing
Execution Plan
Help with Cyber Week preparations
Fulfill rollout goals

Now it’s time to take our initial goal of rolling out Distributed Tracing, our promised contribution to the Cyber Week project (our sponsor project), and the points we wanted to cover during the rollout, and come up with a concrete execution plan that would bring together all three.



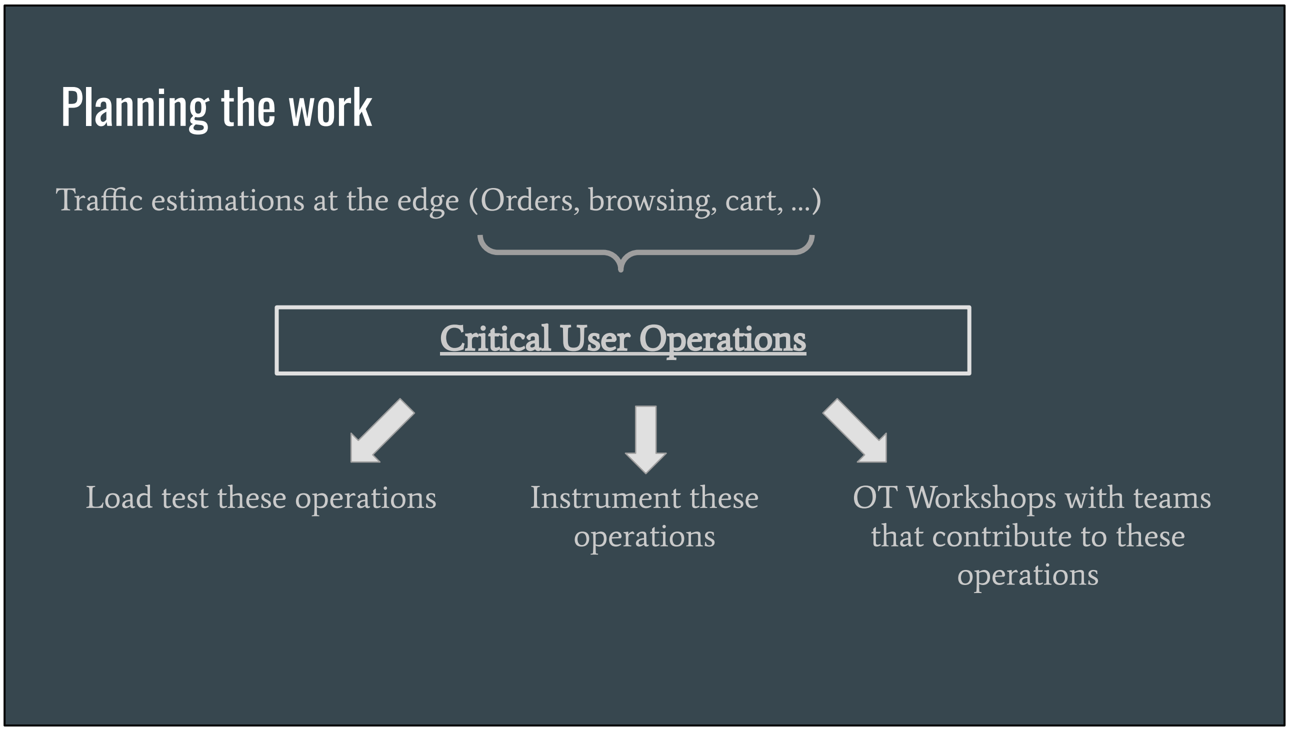
## Planning the work

Key ingredient: Instead of instrumenting applications, **we decided to instrument entire customer operations.**

Why the change to operations?

1. Load testing **Critical User Operations** for Cyber Week
2. OpenTracing's guidance: **"Crawl, Walk, Run."**

The ingredient that was missing in our plan came after a change in the way we looked at instrumentation. We changed the goal from instrumenting applications, to instrumenting customer operations.



## Planning the work

Traffic estimations at the edge (Orders, browsing, cart, ...)

**Critical User Operations**

Load test these operations

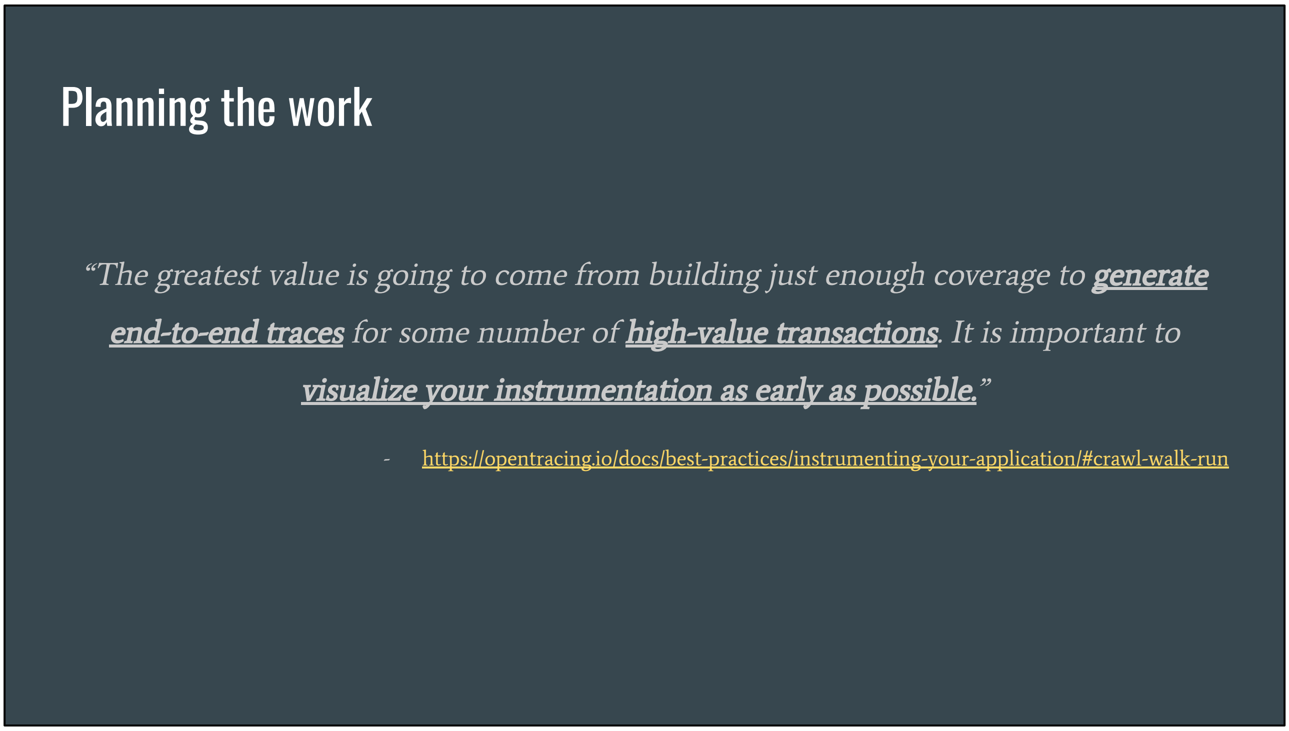
Instrument these operations

OT Workshops with teams that contribute to these operations

Cyber Week gave us the framing we needed to put in practice what we planned for the rollout.

As mentioned before, we had rough traffic estimates at the edge (number of orders, customers browsing the catalog, adding articles to the cart, …)
Together with experienced engineers, we mapped those estimates to concrete operations in the online store, and from there created a list of critical user operations.

For the load tests we would hit the different parts of the platform that served those critical user operations.
We used that list to plan the operations to instrument, and, through that, the teams to run the workshops with.
We chose the teams that developed and operated the services that contributed to those critical operations.

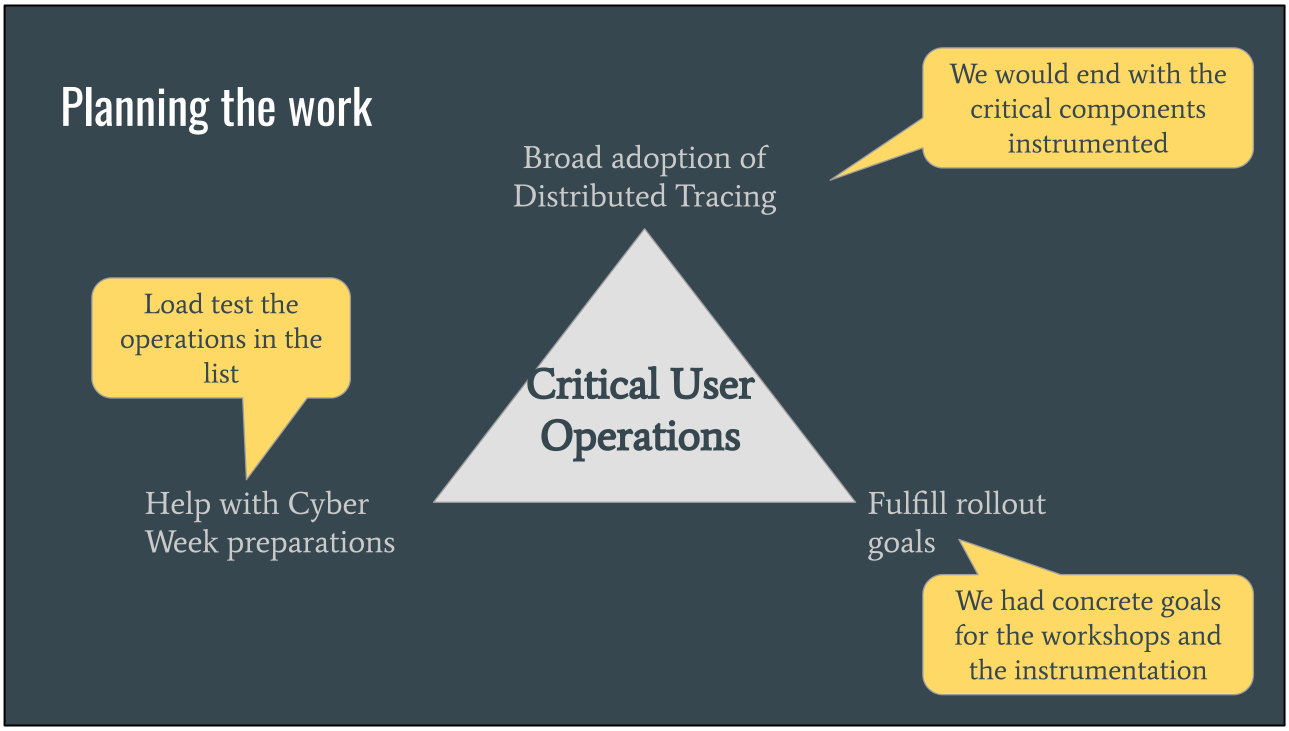


## Planning the work

*"The greatest value is going to come from building just enough coverage to **generate end-to-end traces** for some number of **high-value transactions**. It is important to **visualize your instrumentation as early as possible.**"*

- https://opentracing.io/docs/best-practices/instrumenting-your-application/#crawl-walk-run

OpenTracing also has a best practice for instrumentations named "Crawl, Walk, Run". From the documentation: "The greatest value is going to come from building just enough coverage to **generate end-to-end traces** for some number of **high-value transactions**. It is important to **visualize your instrumentation as early as possible**."
"Generate end-to-end traces" → instrumenting an operation end to end.
"high-value transactions" → or critical operations
"visualize your instrumentation as early as possible." → which fit perfectly with our guiding principle for instrumentation

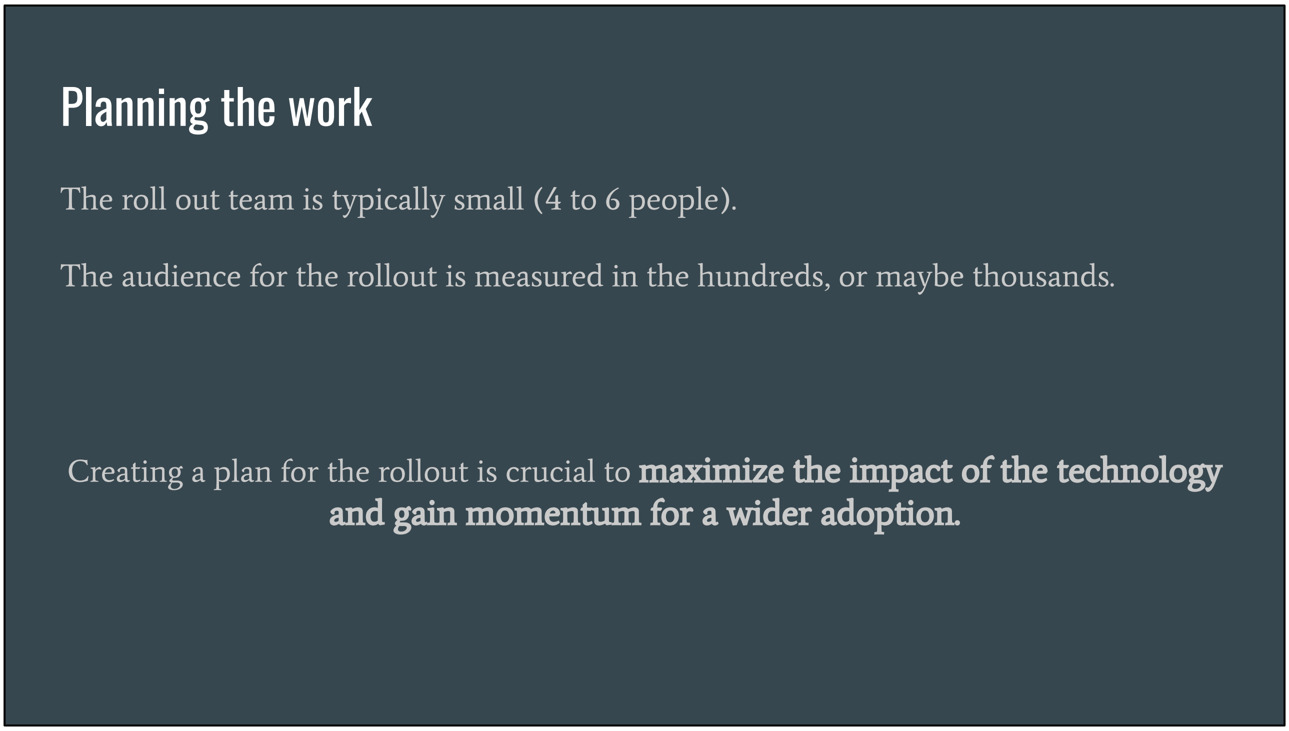


## Planning the work

Broad adoption of Distributed Tracing
We would end with the critical components instrumented
Load test the operations in the list
Critical User Operations
Help with Cyber Week preparations
Fulfill rollout goals
We had concrete goals for the workshops and the instrumentation

I cannot overstate how fundamental that selection, and its criteria, was to the success of the project. It’s what tied everything together.

The decision to focus on the critical user operations allowed us to work on something more concrete, achievable, aligned with the Cyber Week project, and through that it gathered support from management.



## Planning the work

The roll out team is typically small (4 to 6 people).

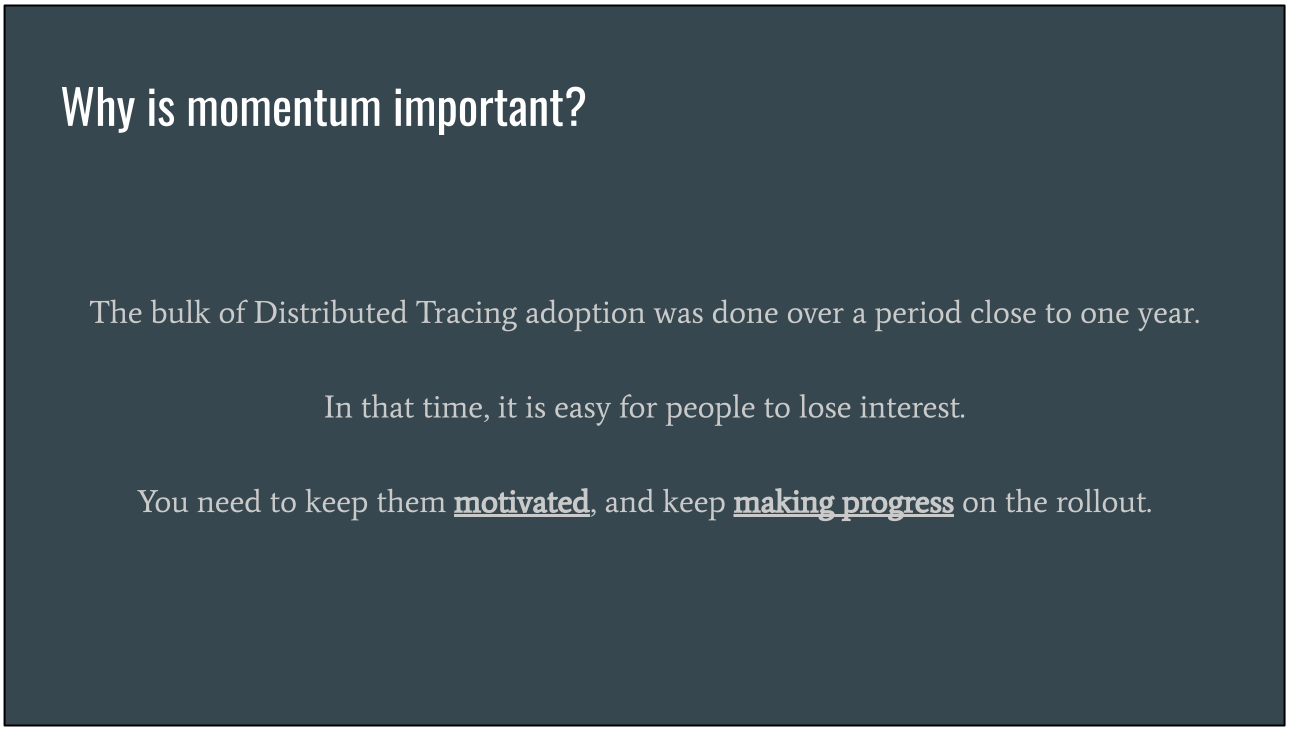
The audience for the rollout is measured in the hundreds, or maybe thousands.

Creating a plan for the rollout is crucial to **maximize the impact of the technology and gain momentum for a wider adoption.**

That list is what allowed a small team to have such a big impact.
Typically, the team in charge of such rollouts is fairly small. In our case there were 6 people working on the rollout, with a team of the same size handling the tracing infrastructure.
Those four engineers could not, in a timely manner, instrument all services.
Regarding the training, each workshop was 3-4 h long. Even by training several teams together, it would still take a long time to train every single team.

“Full speed ahead!”

We had our foot on the door. We kick started the rollout. But it was still a large org. We needed to get those operations instrumented in time for Cyber Week. We needed momentum.



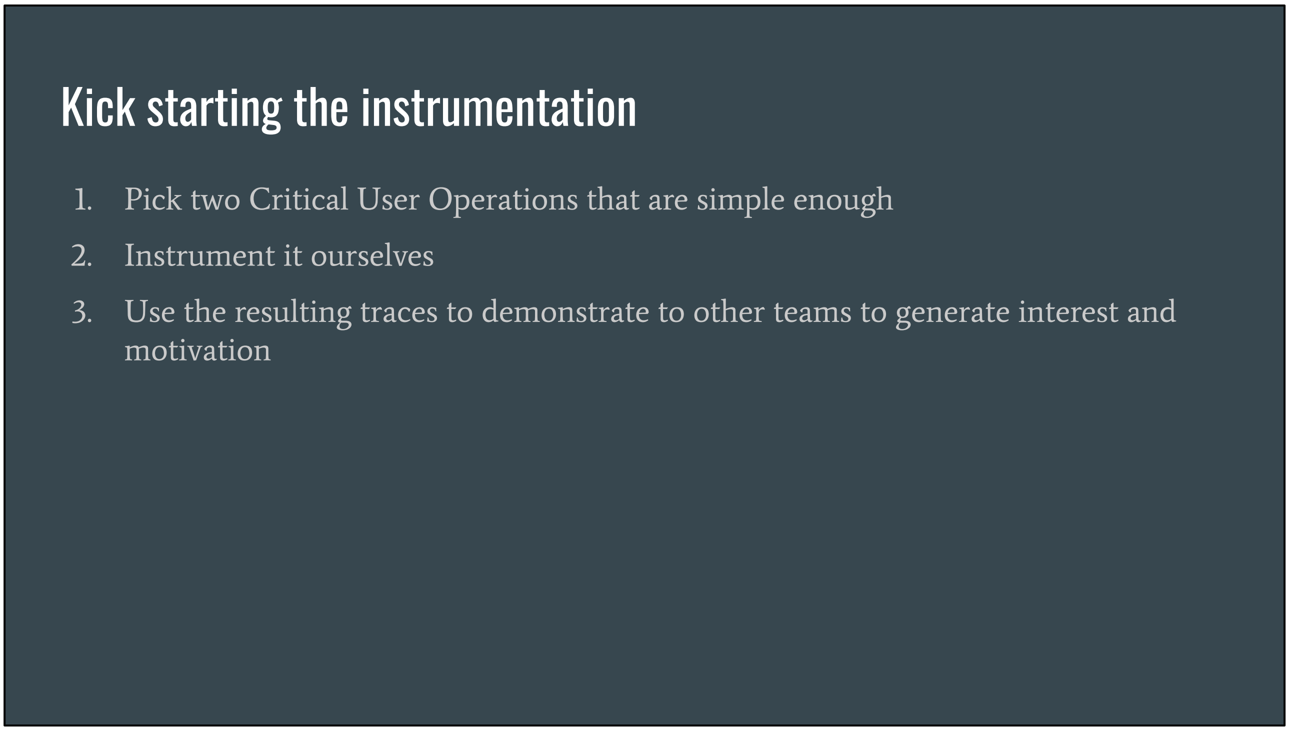
## Why is momentum important?

The bulk of Distributed Tracing adoption was done over a period close to one year.

In that time, it is easy for people to lose interest.

You need to keep them **motivated**, and keep **making progress** on the rollout.

We needed to keep people engaged. This kind of effort is typically done over a long period of time. It is easy for people to lose interest. You need to keep them motivated, and keep making progress on the rollout.



## Kick starting the instrumentation

1. Pick two Critical User Operations that are simple enough
2. Instrument it ourselves
3. Use the resulting traces to demonstrate to other teams to generate interest and motivation

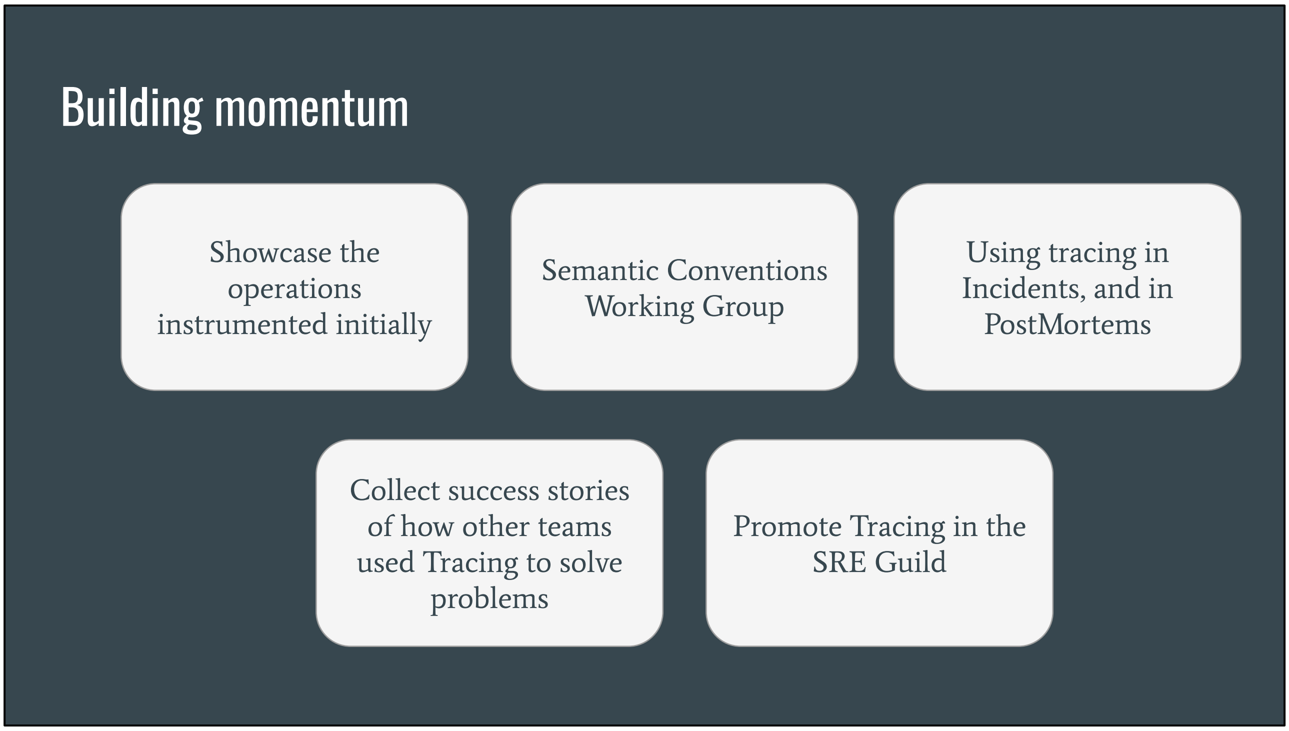
The change to instrumenting critical business operations did not mean that the rollout team could single handedly instrument them all. Those operations still passed through many services. And with different tech stacks.
But we did want to have something to show teams. Some traces from those critical user operations that others could look at to generate interest, and motivation.

So, from the list of critical user operations we selected two that would be critical enough, but also simple enough that we could instrument ourselves.
Having those examples would help with promoting Distributed Tracing to other teams, as they could see concretely what they would get out of their efforts with examples from our own systems.

## Kick starting the instrumentation

1. Pick two Critical User Operations that are simple enough
2. Instrument it ourselves
3. Use the resulting traces to demonstrate to other teams to generate interest and motivation

Fun fact: Although we proposed to those teams that we would do the instrumentation ourselves, and they only had to review the code changes, it wasn't long before they took over, excited with the promise of improved observability.



## Building momentum

Showcase the operations instrumented initially

Semantic Conventions Working Group

Using tracing in Incidents, and in PostMortems

Collect success stories of how other teams used Tracing to solve problems

Promote Tracing in the SRE Guild

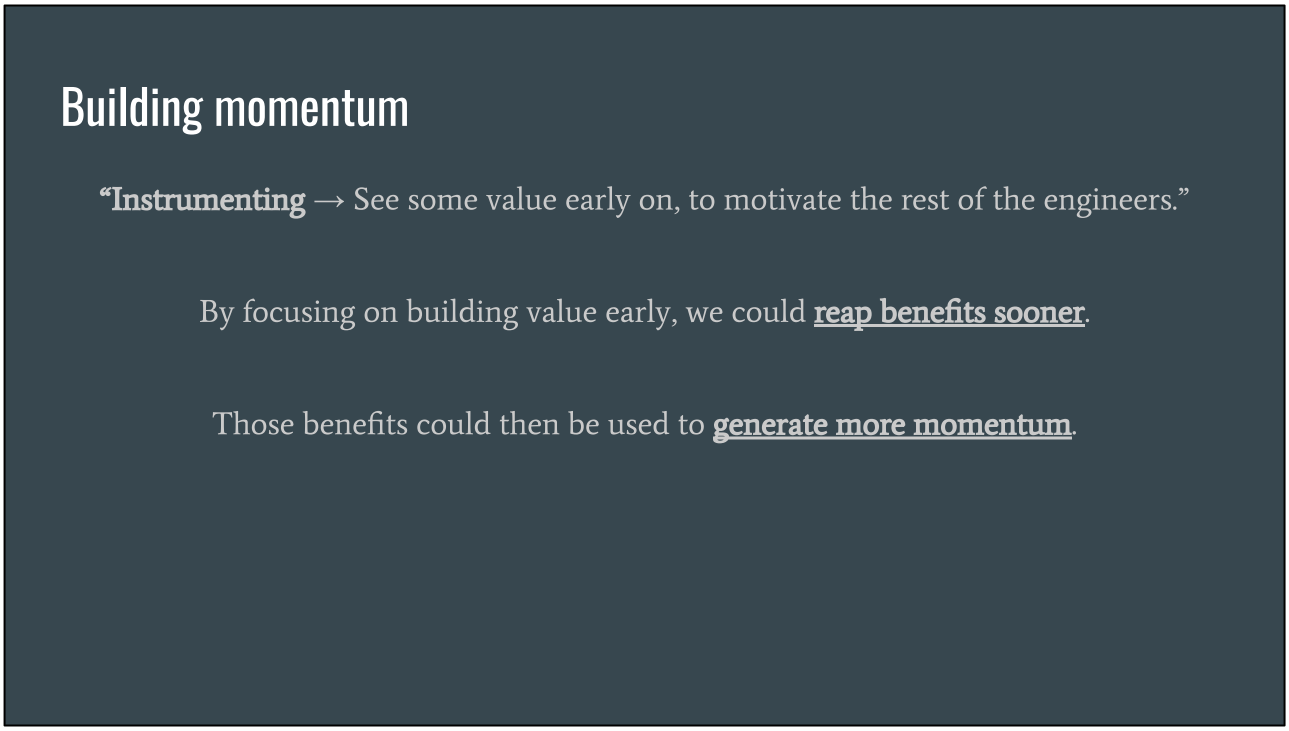
Instrumenting those initial operations was already a big help. With those operations instrumented we could go to other teams, and rather than promises like "Tracing will be great for you", we would say "Look how great it looks like for these operations".

Teams often asked how Tracing’s improved observability could actually helps them. To provide concrete answers we started collecting success stories. Episodes of how other teams in the org used Distributed Tracing to help fix problems, or even identifying problems they didn't even know they had. How those episodes improved latency, helped optimize resource usage, and fixed latent errors.

We also created a Semantic Conventions Working Group to build on top of OpenTracing’s semantic conventions, defining common tags and best practices for our specific use cases.

The SRE team also ran the SRE Guild. The guild was often used to promote tracing. For example, we would often invite the writers of some of those success stories to guide the audience through those experiences.

Incident handling was a key use case, so during incidents, in our role as Incident Commanders, we would often ask for traces to demonstrate the issue we were facing at the time. In postmortems we would highlight situations where tracing could have been instrumental to solving the incident, and recommend as an action item of the postmortem instrumenting the services involved.

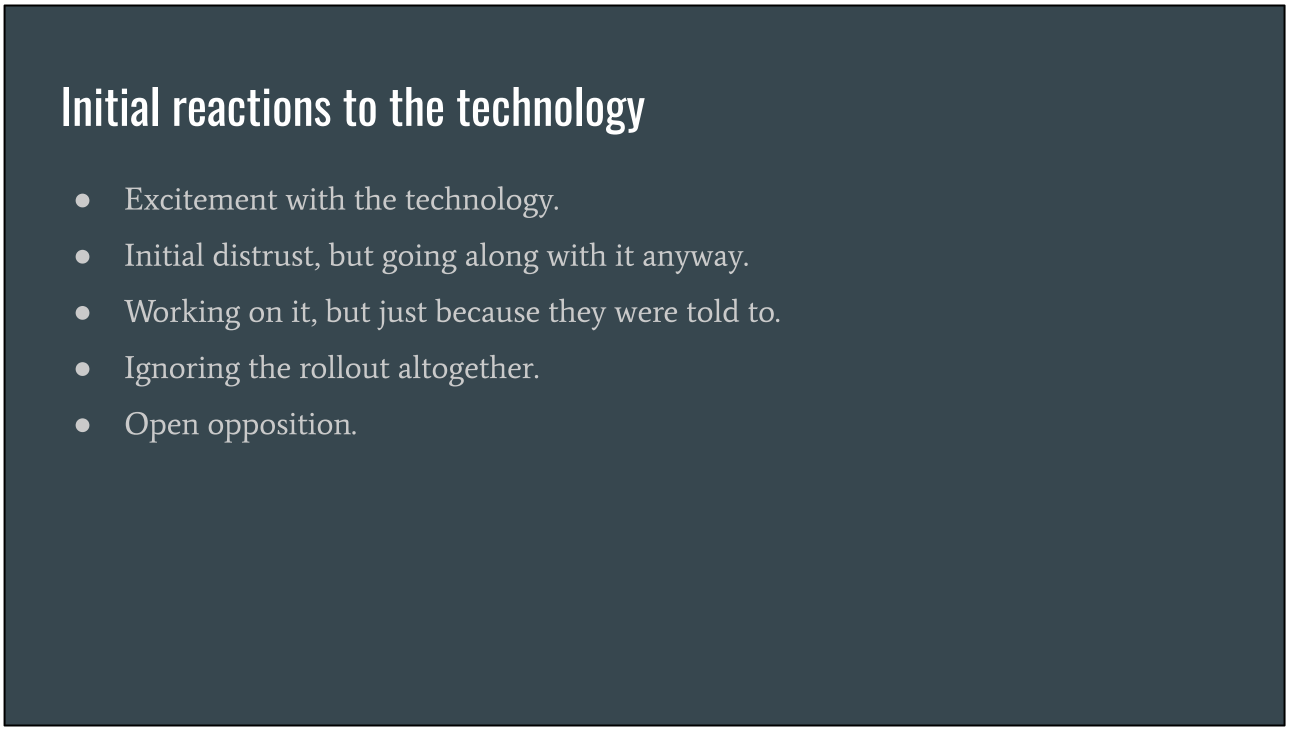

## Building momentum

"**Instrumenting** → See some value early on, to motivate the rest of the engineers."

By focusing on building value early, we could **reap benefits sooner**.

Those benefits could then be used to **generate more momentum**.

Going back to the three aspects of the rollout, because we set out to generate value early with our instrumentation approach, we were also able to start collecting benefits early from Distributed Tracing.
As coverage increased, processes also started changing to benefit from Tracing. And we used those benefits to feed into generating more momentum.

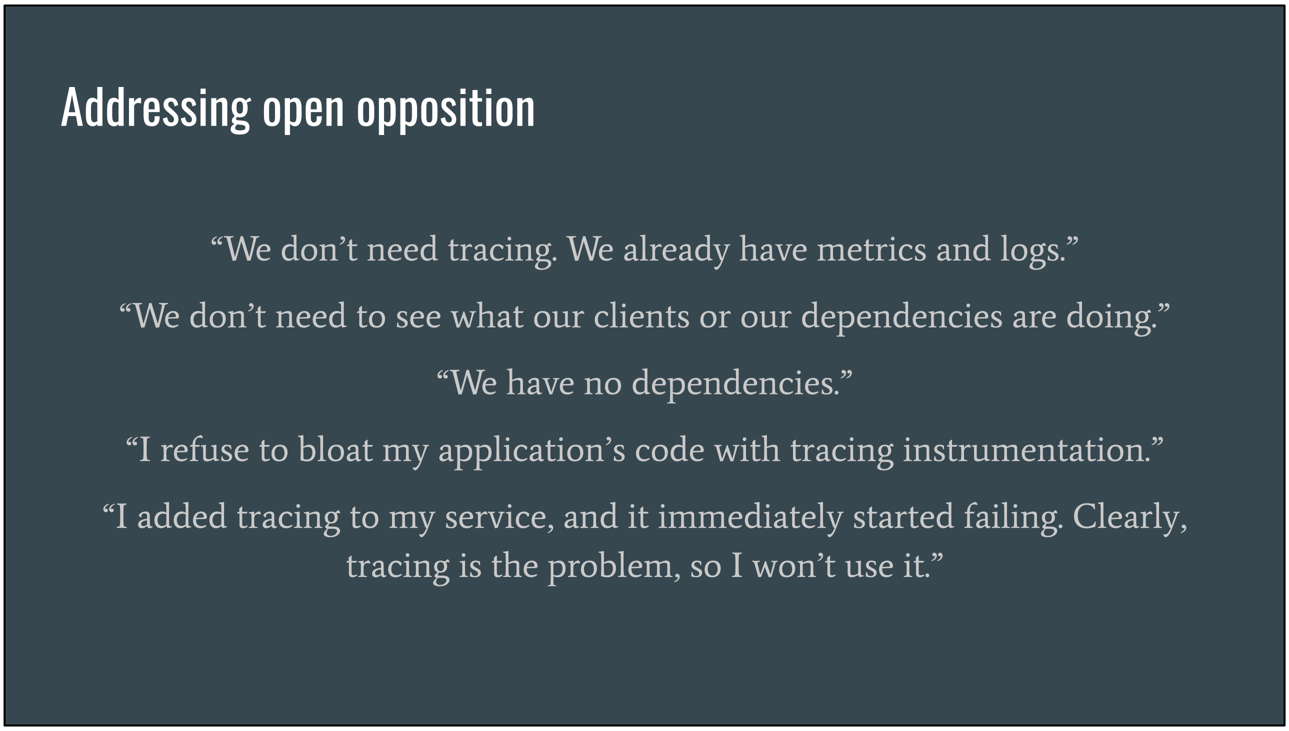


## Initial reactions to the technology

- Excitement with the technology.
- Initial distrust, but going along with it anyway.
- Working on it, but just because they were told to.
- Ignoring the rollout altogether.
- Open opposition.

What was the reaction from the engineering community to all of these initiatives? I think we had a very diverse set of reactions.

You'll notice that not all reactions are that positive. Building and maintaining momentum also means addressing that initial resistance to the technology.



## Addressing open opposition

"We don't need tracing. We already have metrics and logs."

"We don't need to see what our clients or our dependencies are doing."

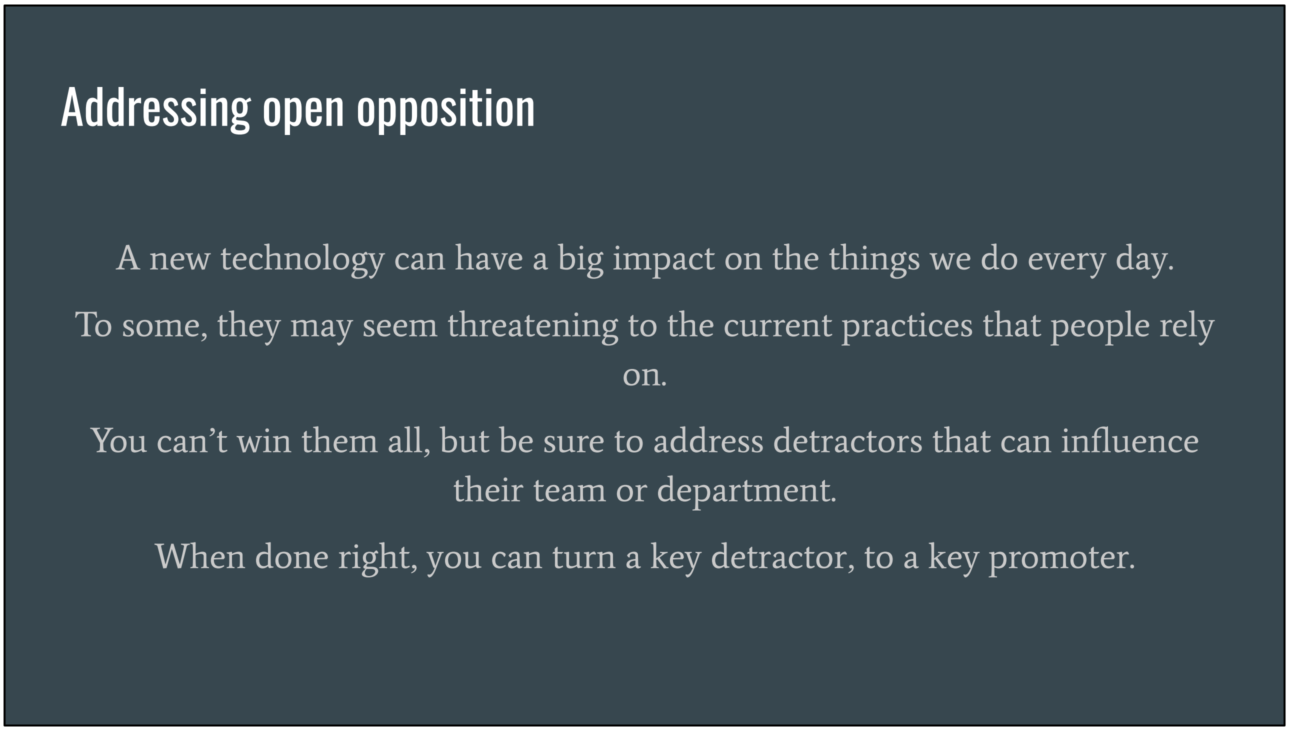
"We have no dependencies."

"I refuse to bloat my application's code with tracing instrumentation."

"I added tracing to my service, and it immediately started failing. Clearly, tracing is the problem, so I won't use it."

But now I'd like to talk about the open opposition we also got.

Some of these quotes showcase really well how we sometimes cling to our tools, pushing aside alternatives that provide more benefits.



## Addressing open opposition

A new technology can have a big impact on the things we do every day.

To some, they may seem threatening to the current practices that people rely on.

You can’t win them all, but be sure to address detractors that can influence their team or department.

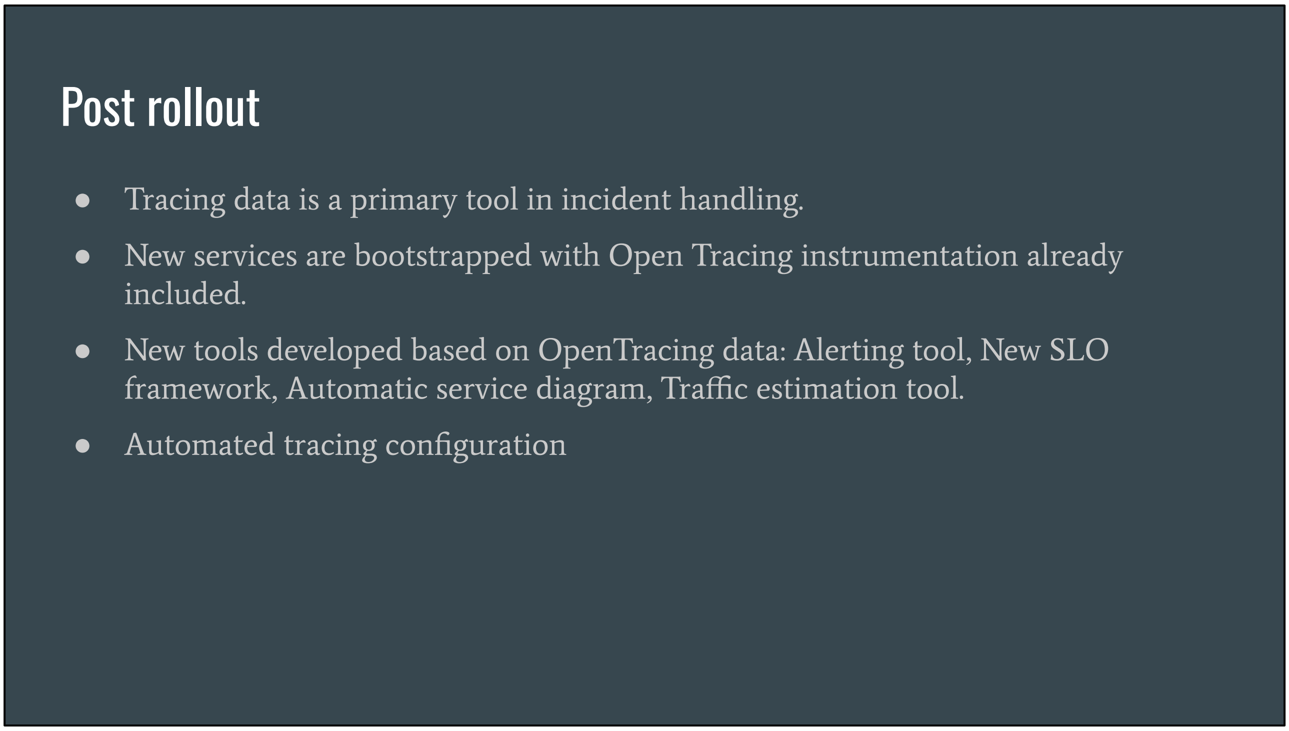
When done right, you can turn a key detractor, to a key promoter.

We did not reach out to every single team, or person that was opposed to Distributed Tracing. But we did engage with some of them. Typically because they were blockers to instrument critical operations, or at least part of those operations.

What is important is to recognize that new technologies can have a big impact on the things we do every day. To some, they seem threatening to the current practices that they rely on. It's like you are taking away the very thing that allows them to be effective at their job, and replacing it with something unknown. As with all things, change can be scary.
Do not ignore that. Do not assume that those are some random individuals too set in their ways. Some of those random individuals can actually have quite a big influence within a team, or department. Others you can afford to give them some more time, so give them that time.

## “Are we done yet?”

The rollout was ‘attached’ to the cyber week project, so when that was done, so was the rollout. We did get the critical user operations instrumented in time for the load tests, which were a success, and we could rely on tracing to monitor the performance of those operations during the event itself.



## Post rollout

- Tracing data is a primary tool in incident handling.
- New services are bootstrapped with Open Tracing instrumentation already included.
- New tools developed based on OpenTracing data: Alerting tool, New SLO framework, Automatic service diagram, Traffic estimation tool.
- Automated tracing configuration

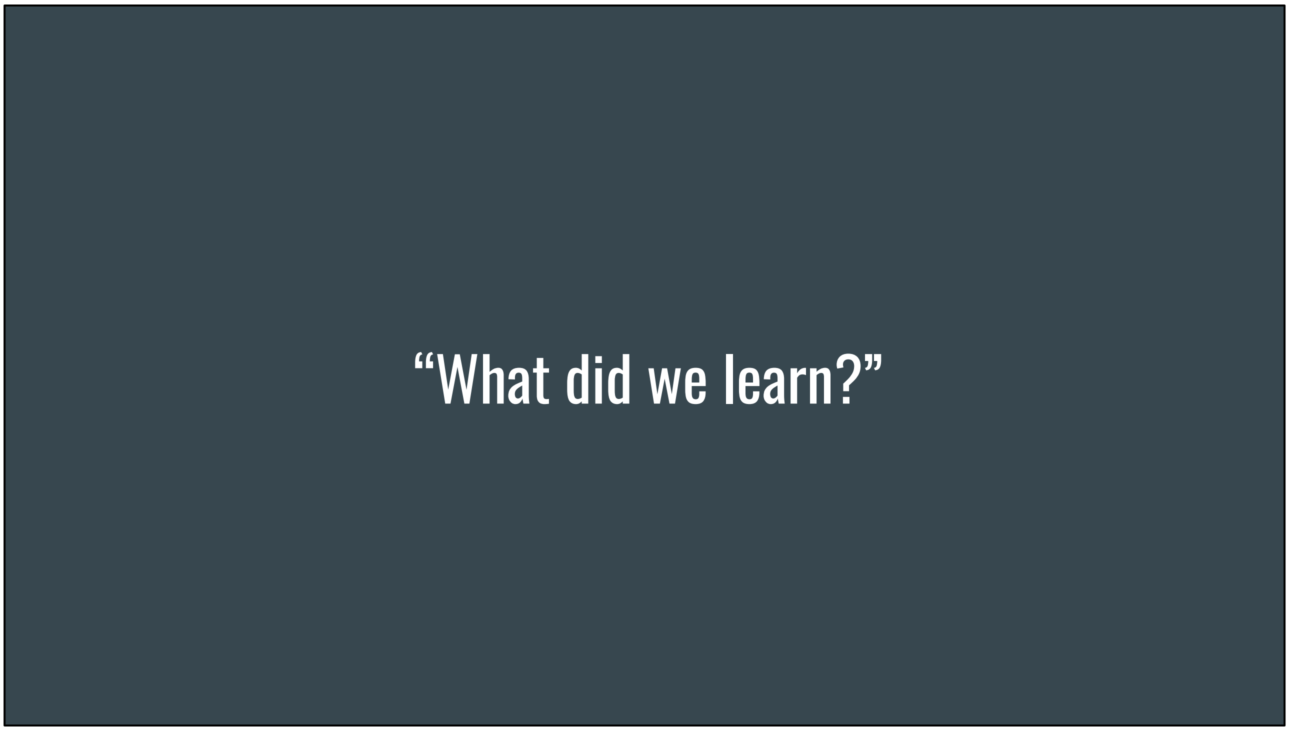
By now we are past the adoption phase. Distributed Tracing is well established in the company. Of course, there is still a long tail of adopters, but more often than not, you'll find tracing wherever you will go.

Tracing data became the primary tool in incident handling.

New projects are bootstrapped with tracing instrumentation already included.

By accessing the tracing data programmatically we were able to develop tools that helped with some operational challenges. We developed an alerting and triaging tool, developed a new SLO framework, used tracing data with other tools to create a service diagram, developed a traffic estimation tool to help load testing, and capacity planning.

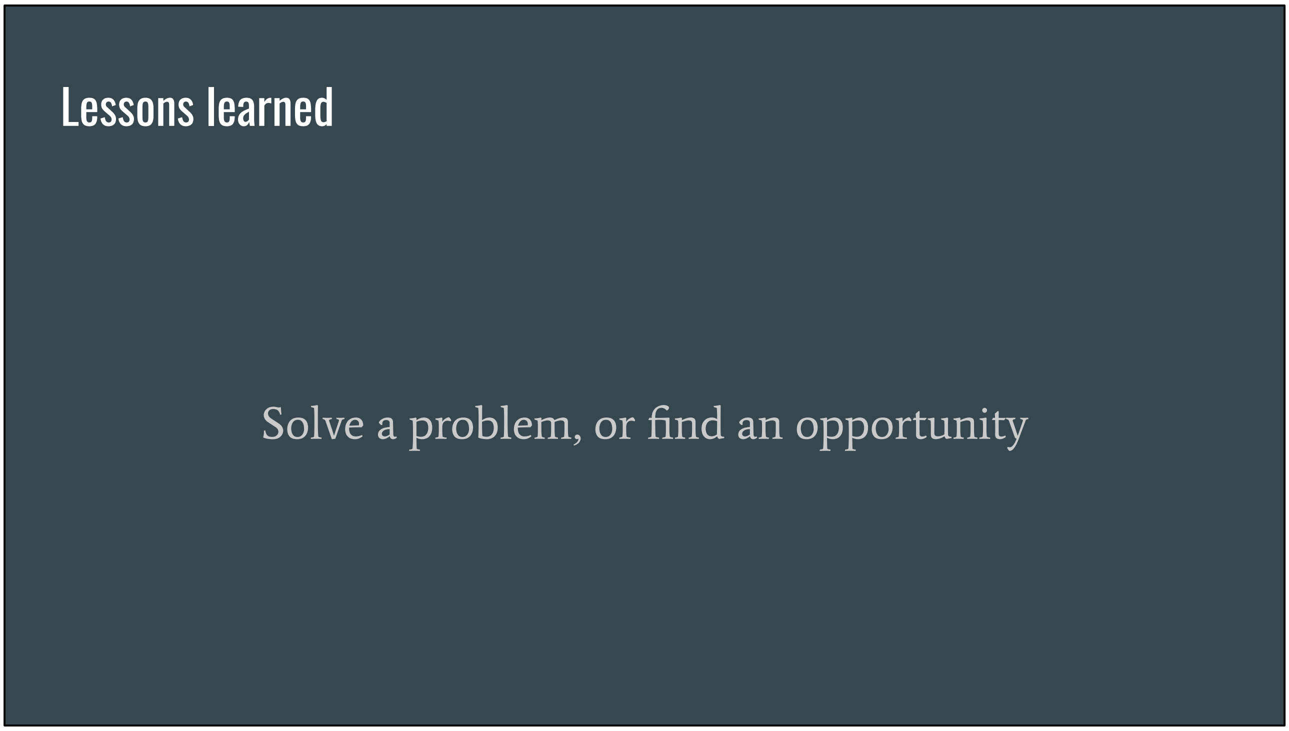
We also kept our ears open to feedback from developers, which is how we learned that configuring the Tracer was still annoying. We developed self configuring libraries for the main programming languages.



## “What did we learn?”

Personally speaking, this was a very exciting project to work on. It was full of different challenges, and the end result was really rewarding to see. We changed the way an entire engineering organization worked, and we built some very cool stuff in the process.

But I’d like for you to leave this talk with something more than storytelling, so let’s look at what learnings we can take from this story.

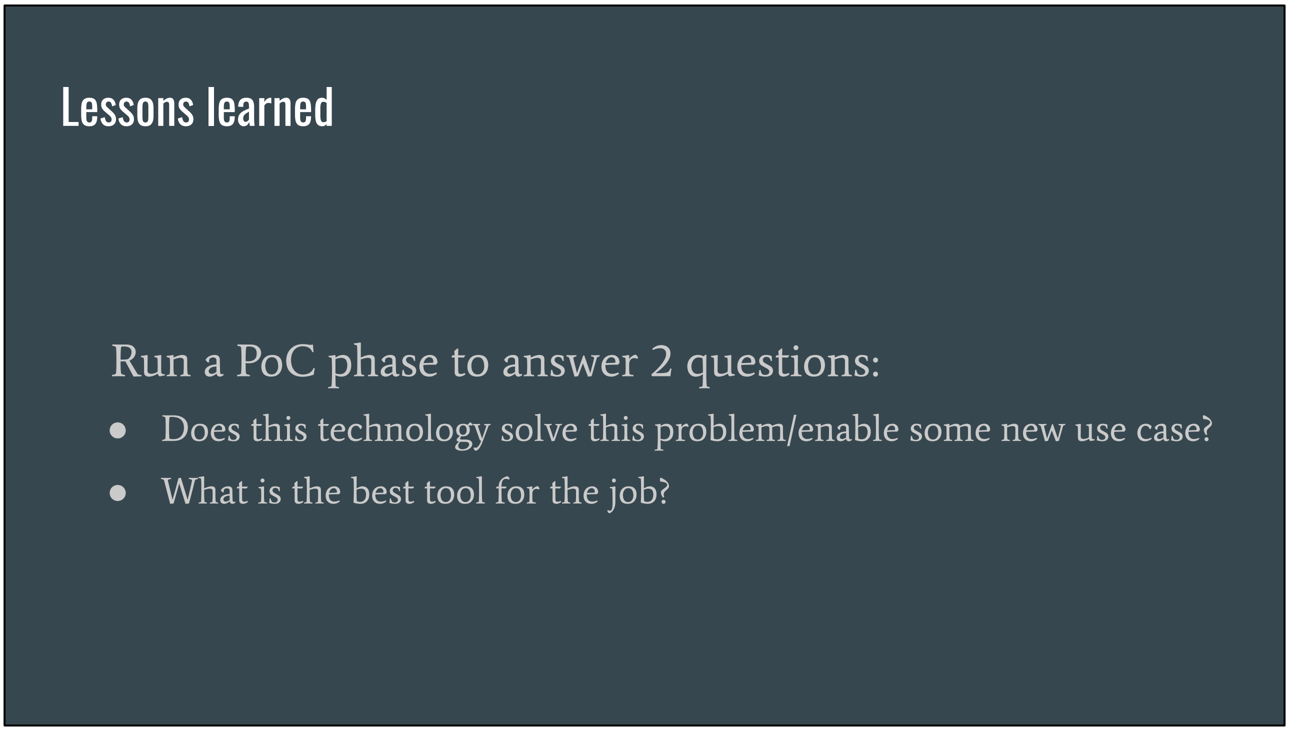


## Lessons learned

Solve a problem, or find an opportunity

Solve a problem, or find an opportunity

- Start from this, and the purpose for whatever technology you are trying to onboard will always be clear. To you, or to the people you are bringing the technology to.
- In our case, the problem was observing a distributed system. But new technologies don't have to solve problems exclusively. They can also be onboarded to leverage new business opportunities.



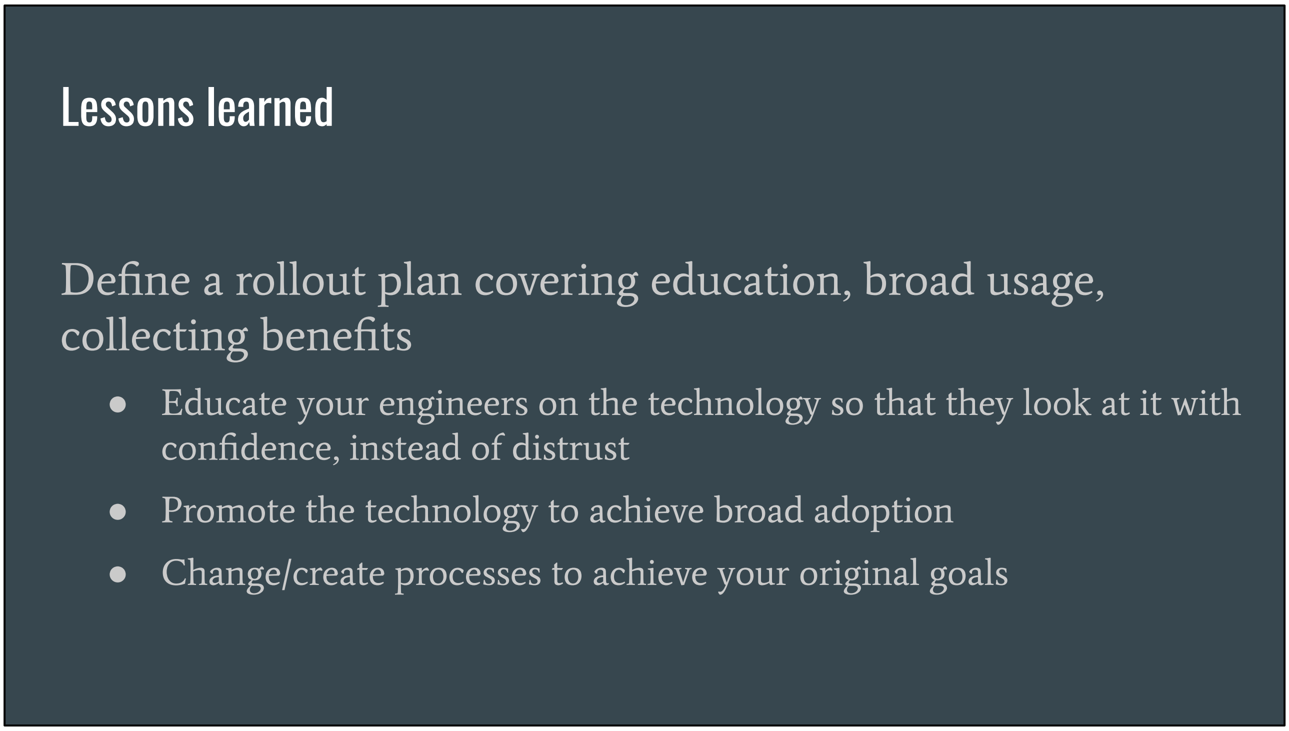
## Lessons learned

Run a PoC phase to answer 2 questions:

- Does this technology solve this problem/enable some new use case?
- What is the best tool for the job?

Run a PoC phase to answer 2 questions:

- Does this technology solve this problem/enable some new use case?
- What is the best tool for the job?
- The decision to build or buy depends greatly on the culture and strategy of an engineering org, of course. If you’re still on the fence, though, remember to set clear criteria on which to evaluate whatever options you consider
  - Cost, scale, staffing, talent, features, customization, etc…



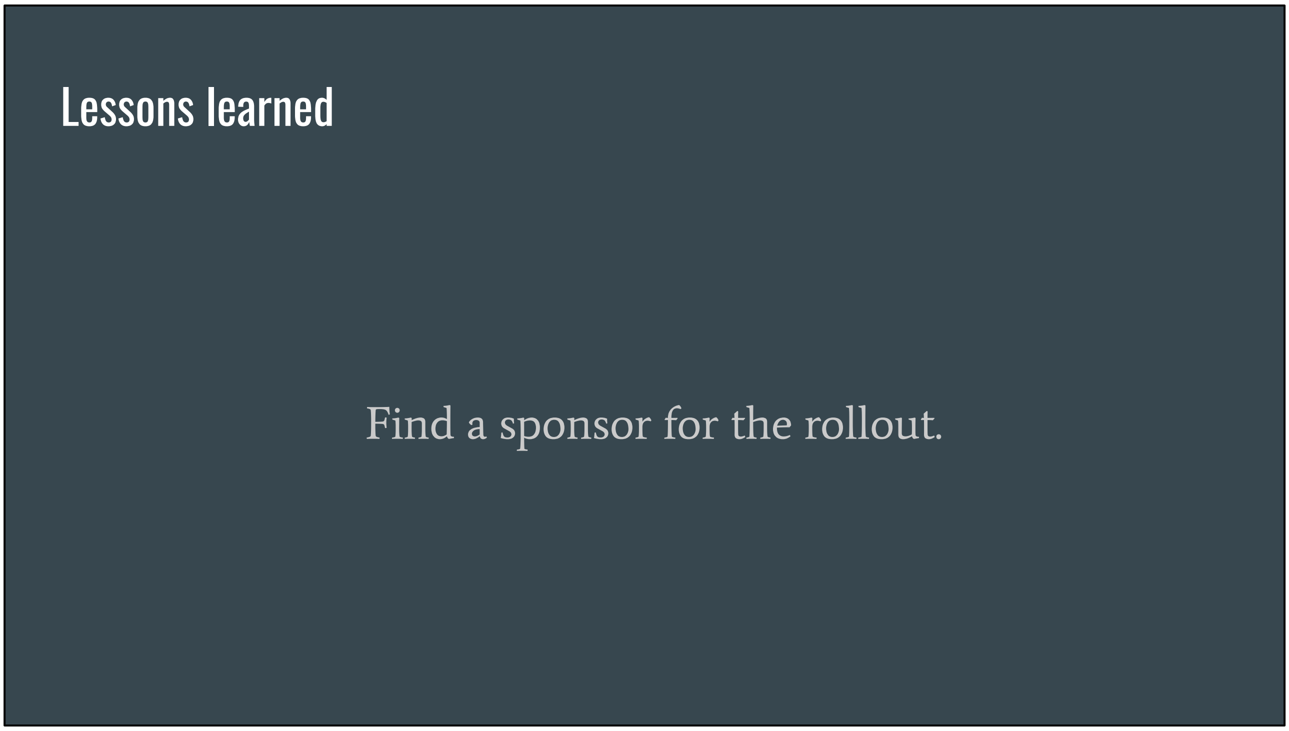
## Lessons learned

Define a rollout plan covering education, broad usage, collecting benefits

- Educate your engineers on the technology so that they look at it with confidence, instead of distrust
- Promote the technology to achieve broad adoption
- Change/create processes to achieve your original goals

Rollout plans should be more than just ticking a box that a given technology is somehow available in some team, or is being used in X services. That being said, remember to define a rollout plan covering education, broad usage, collecting benefits

- Educate your engineers on using the technology so that they look at it with added confidence, instead of distrust
- Promote the technology to achieve broad adoption
  - Find some seeds in the org that you can later use to showcase the benefits of the technology
  - Scale your efforts by finding other promoters in different parts of your company
  - Advertise success stories to help build and maintain momentum.
- Change/create processes to achieve your original goals
  - Your work is not done when something is everywhere, but rather if it is being used in enough places, solving the original purpose
  - Remember your initial goal (problem solving, or new opportunity) and make sure to make changes where needed so that your company can reap the benefits of the investment

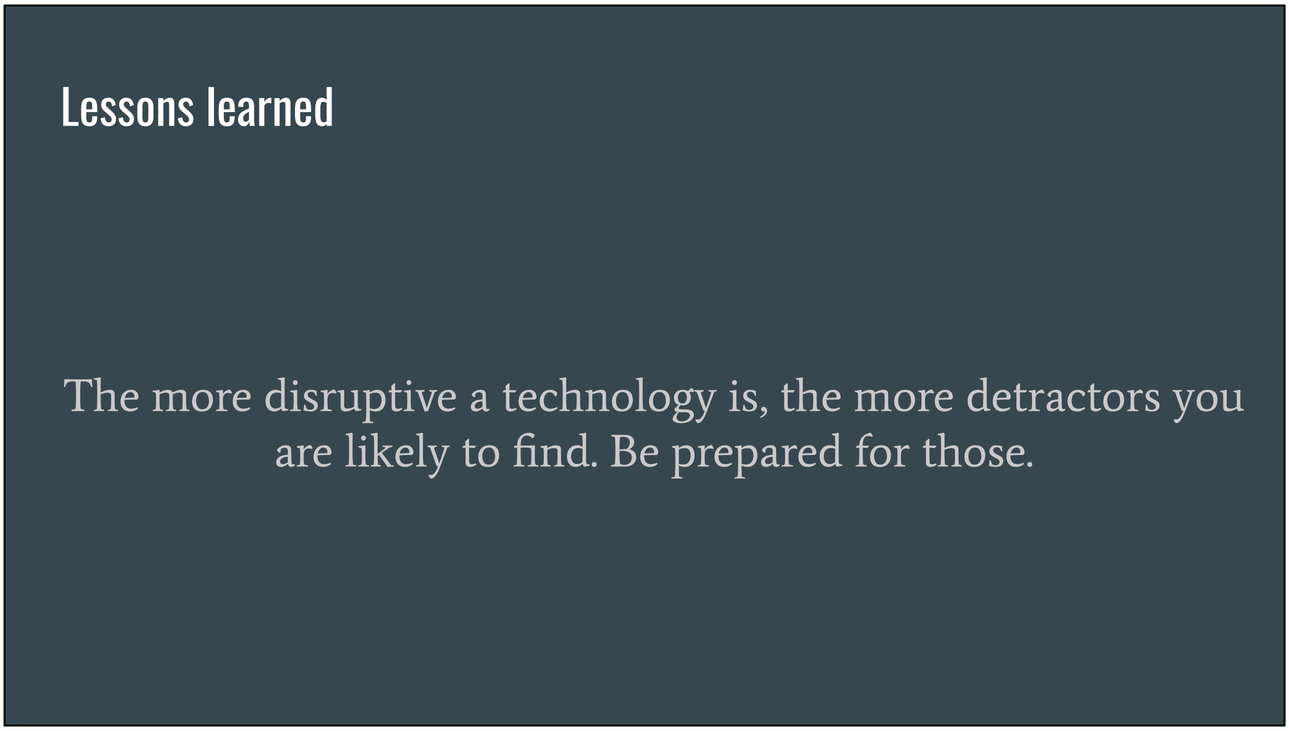


## Lessons learned

Find a sponsor for the rollout.

Find a key sponsor for the rollout.

- This will also help validate the promised benefit the technology brings to the org.
- You may have thought you already had that when you passed the PoC phase, but if you are coming from a central team (which was our case) in a large enough org, there are many competing priorities and projects.
- Your key partner may be a high priority project, or one or more managers that believe in the benefits promised with the new technology, and will support the adoption of the technology in their orgs.

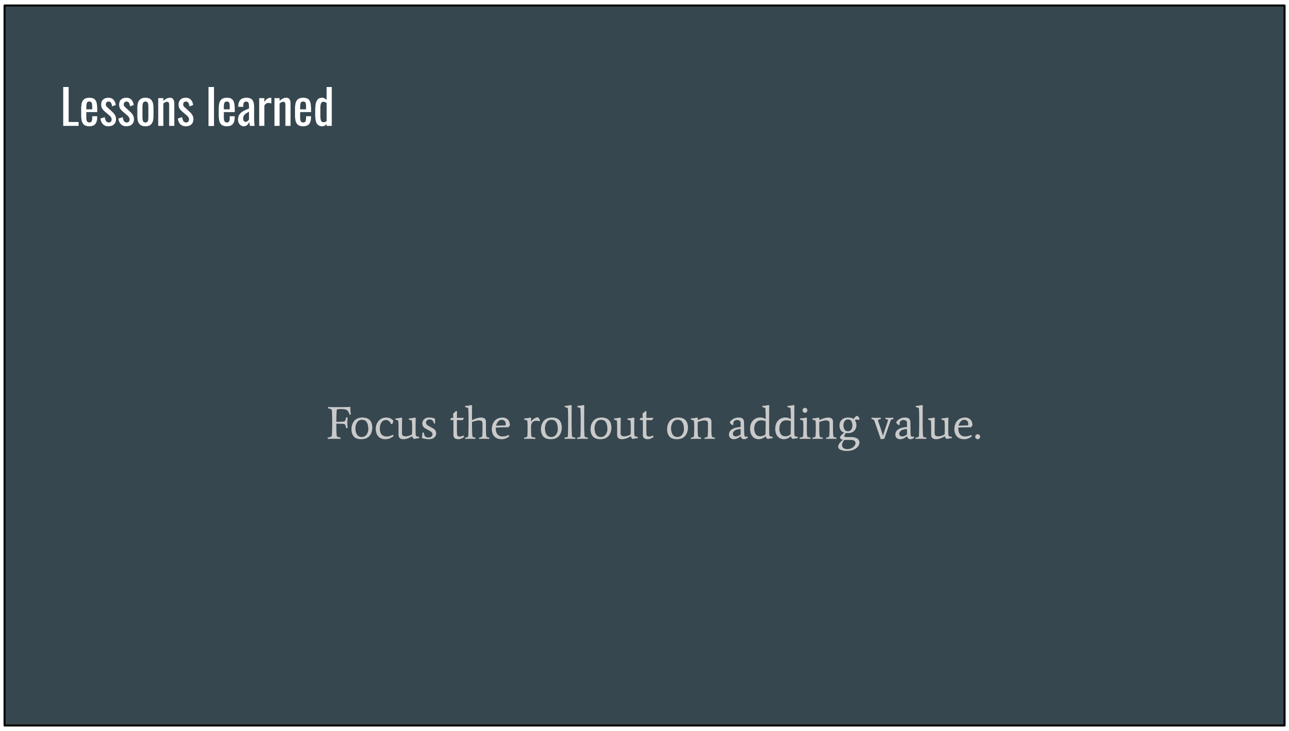


## Lessons learned

The more disruptive a technology is, the more detractors you are likely to find. Be prepared for those.

The more disruptive a technology is, the more detractors you are likely to find. Be prepared for those.

- You can’t reach all of them, but acknowledge and address elements in key areas of the org.
- When done right, those will eventually become your main promoters.



## Lessons learned

Focus the rollout on adding value.

Focus the rollout on adding value

- I keep coming back to this, but this was the mindset throughout the whole project.
- Don't tell teams "You have to do this". Tell them "You can use this to solve those problems you have."
- If you're out to solve people's problems, they are a lot more likely to adopt whatever it is you are proposing.
- If you're just pushing it to them, then, sure, you may get the coverage you're aiming for, but you will not be able to collect the same benefits.
- By focusing on adding value, it also becomes more easy to motivate yourself.

## Thank you!

Pedro Alves @ SRECon 2023