devtron

Building the Business Case for Developer Experience

devtron.ai

Introduction

As a DevOps or platform engineering lead/manager, you should be familiar with this situation: Your engineers are inundated with requests to maintain environments and manage integrations, particularly from those onboarding to your organization or Kubernetes for the first time. They've tried to build a complex CI/CD pipeline to catch errors early and deploy released on time, but their constant backlog has led to a daisy chain of ungoverned custom scripts.

In response, you decide to templatize all your Kubernetes configurations to strictly enforce how developers use the Kubernetes environment going forward. In the aftermath, your DevOps team indeed finds their workload lightened, but at the expense of your downstream developers, who lose hours struggling to adapt to new paradigms that affect their release velocity and quality control.

Instead of an investment that returns long-term, compounding impact for your customer—the internal developer—you've implemented a point solution that offers a siloed benefit at best, or a net decrease in release velocity and quality at worst.

Instead, you might have considered building a business case around investing in developer experience (DX), the overall efficiency and effectiveness of the tools and frameworks of your software development lifecycle (SDLC). After all, a great DX ultimately leads to better DevOps and developer productivity, which is what every executive is looking for today. We're not going to focus on what DX is or the best practices you might build your unique use case around—for that, take a look at our companion eBook, **Unleashing Developer Productivity: Leveraging Developer Experience As Your Secret Weapon**

Instead, let's focus on what you need to create a compelling, approval-ready business case around DX. We'll focus on starting the right conversations to reveal pain points, designing your solution, discovering your "hook," and ultimately pitching the impact of DX, whether that's faster development, smoother learning curves, higher code quality, or beyond.

devtron

Part 1: Set up the conversation with stakeholders

When developing a business case for DX, your stakeholders are practitioners. Most importantly, your developer customer, followed by your DevOps or platform engineers who can implement the toolkit and processes to make their lives easier.

Start by getting these folks into the same room for open-ended conversations. Ask them where they are and aren't satisfied with the status quo, what challenges they face, and where they see opportunities for improvements. Existing research and surveys provide helpful waypoints for asking the right questions:

Top Priorities Driving Investment in DX

47% of developers and engineers are overworked

31% struggle with inefficient processes

83% are concerned about software reliability

55% were frequently delayed due to inefficient processes

Consequences of Not Investing in DX

87% aim to improve developer productivity

85% want to improve the software supply-chain practices and governance

85% are trying to speed up release cycle times

83% want to onboard new developers faster

Tip: When engaging with developer stakeholders, remember to operate from a place of listening, discovering, and collaborating. This moment is not about informing them or dictating how they should work, but figuring out how you can best use the talent within your team to deliver solutions that solve their problems.

devtron

These conversations are not the only data you need to gather, but they should provide enough context to start developing the "hook" for your entire business case. What is the headline problem your organization can solve with DX? What returns can you promise, or costs of maintaining the status quo, will grab leadership's attention and convince them to commit?

Let's turn back to the story we started with as an example of a compelling hook.

Instead of stampeding to templatized Kubernetes configurations, you engage developer stakeholders, who complain about the CI/CD pipeline, which creates downtime during execution and doesn't include intuitive debugging and testing tools. They know they're releasing slower and rolling back more deployments than industry average, leading you to the compelling conclusion that better DX around quality and delivery could save large—and valuable—chunks of developers' time.

Before

We could save each DevOps/platform engineer 1 hour/day from responding to developer requests or maintaining the software delivery pipeline, freeing them to work on production stability. This results in 560 working hours per year (assumes 2 engineers) or ¼ of an additional DevOps engineering resource.

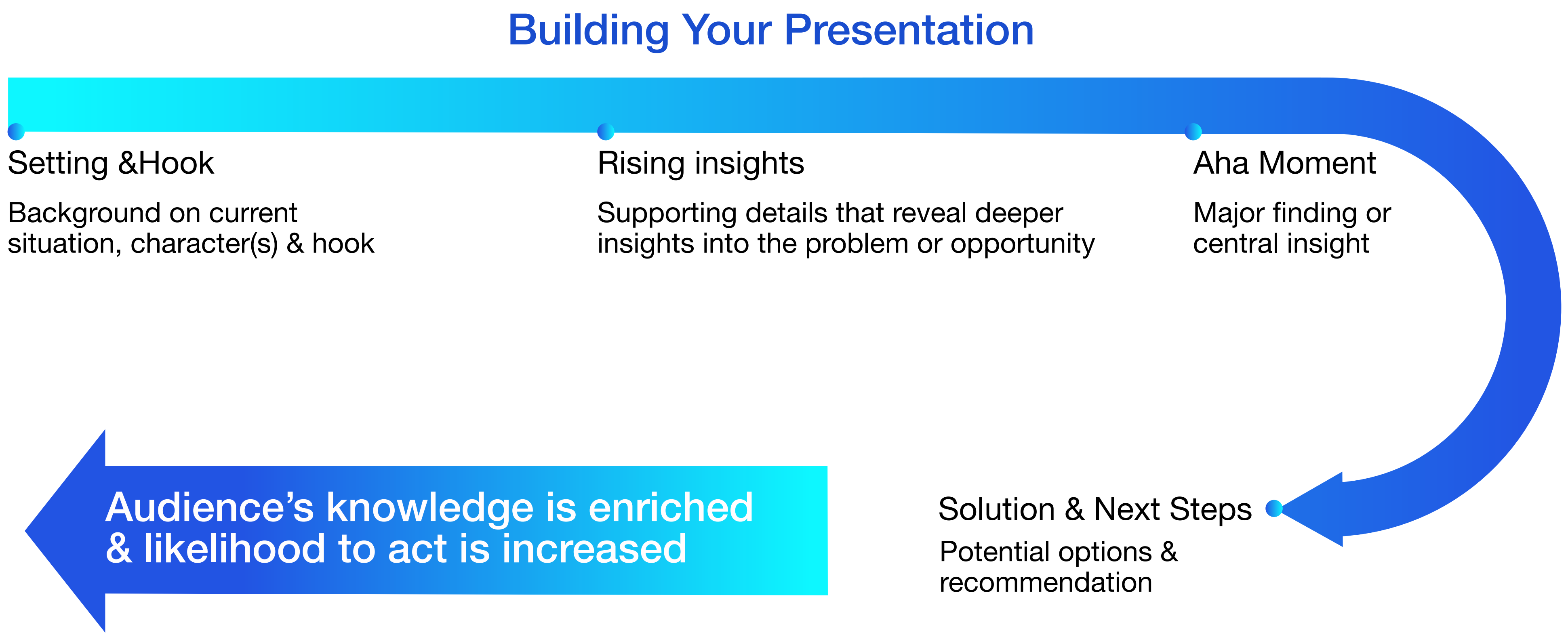
After

We could save each developer 45 minutes/day by investing in a CI/CD pipeline and testing tools with better DX, resulting in 4,200 working hours per year (assumes 20 developers) — equal to bringing in another 1 ¾ developer resources to focus on writing more business functionality.

The new hook isn't just more relevant to core developer pains—it's also far more understandable and compelling to the results your leadership are most interested in.

Part 2: Compile existing data, potential impact, and your solution

With the first part of your discovery process behind you, and your hook drafted, you've built a reasonably mature business case. You have a problem (a slow and often-faulty release cycle), and a specific goal (a 2X improvement in error-free release frequency), but your story still has gaps your leadership will certainly ask about.


Building Your Presentation
Setting &Hook
Background on current situation, character(s) & hook
Rising insights
Supporting details that reveal deeper insights into the problem or opportunity
Aha Moment
Major finding or central insight
Solution & Next Steps
Potential options & recommendation
Audience's knowledge is enriched & likelihood to act is increased

devtron

Question: What data supports the hook and story behind the investment you propose?

Next, you need to understand where your organization fits against the industry average in the areas you're focused on improving through DX. These data become your levers for crafting your solution and confidently predicting the outcome.

There is no one-size-fits-all template for gathering relevant metrics about DX. Let your stakeholder conversations and in-development hook guide you, with the following as just a sub-section of the possibilities:

Performance metrics, like deployment frequency and change failure rate as defined by DevOps Research and Assessment (DORA), which can identify where your organization compares to industry averages.

Code quality, derived from peer code reviews, which identifies gaps in standards and compliance that could have downstream consequences on quality.

Retention rates, which point to burnout, dissatisfaction, or a lack of career development.

Surveys that can identify trends or problem areas in developer wellbeing.

Project success around cost, feature completion, and customer satisfaction, revealing how well teams collaborate around big-picture goals.

Time to first contribution from a new hire, signaling the complexity of your SLDC and whether folks feel empowered to contribute.

devtron

Question: What happens if we do nothing?

Leaders inevitably seek investments that overcome the risk inherent with any major tooling or process change, and one of the best ways to convince them to commit is by clearly stating what they lose in opportunity cost if they fail to act on time. Based on your conversations and research, you'll want to determine how the status quo impedes their long-term goals.

Are they most concerned about a further slippage of failure rates? Even slower release frequency? Are they worried about increased customer churn due to a lack of innovation and platform improvements, or are they worried about their most talented developers leaving due to intense burnout?

Calculating the cost of doing nothing is where a good business plan transitions from focusing on the customer (the developer) to your audience (leadership). A CTO, COO, and CEO have different perspectives, demands, and levels of technical expertise—make sure you frame your opportunity costs around their awareness and top-level challenges. They might not care about reducing developer downtime, but they will certainly be interested in how that exposes new opportunities, like revenue growth or a faster time to market.

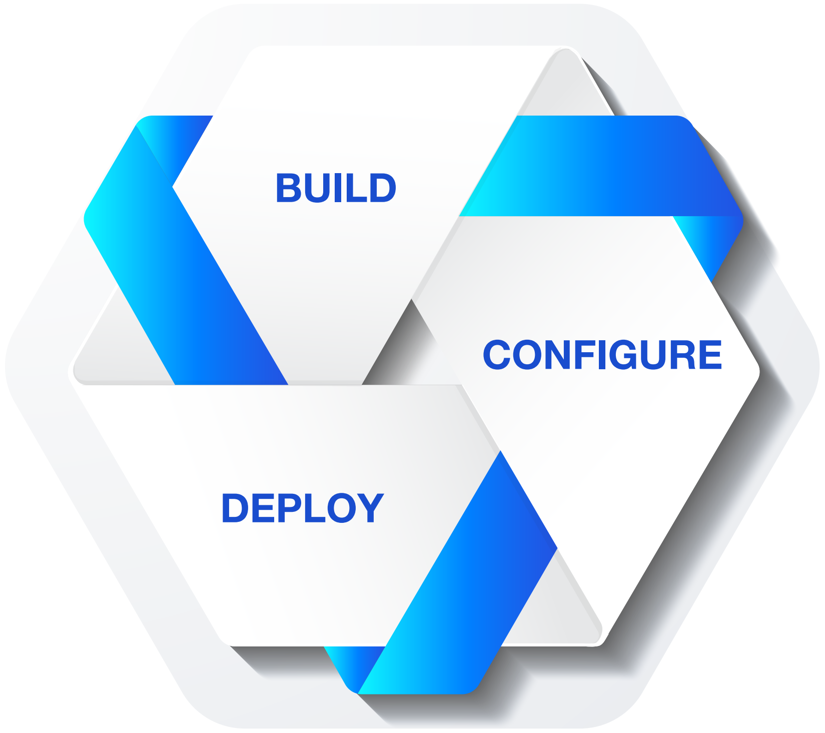
> **Doing something costs something. Doing nothing costs something. And, quite often, doing nothing costs a lot more!**
>
> **Ben Feldman**

devtron

Question: What is your situational assessment?

With an opportunity cost analysis and clarity through data, you're ready to solidify your situational assessment. These come in many forms and different methodologies, which we'll let you explore on your own, but your audience will expect you to answer the following questions at a minimum:

What is the core problem?
Which type of investment will improve or resolve said problem?
What tools and processes will get the job done?
How long will the project take, and who will need to be involved?
What is your predicted outcome?



Continuing the example story of trying to reduce defects and cut back on downtime, you might calculate that not improving DX could cost your organization hundreds of developers hours every year, delaying the delivery of the average project by two weeks at a significant run-on cost. To address the ongoing DX issues, you propose creating a "golden path" of tools for any developer building, configuring, and deploying on your infrastructure, including a streamlined CI/CD pipeline and new tools for end-to-end testing and validating Kubernetes configuration against policy.

Based on your interviews and data collection, you validate your earlier prediction, where this DX-rich golden path will save every developer an average of 45 minutes/day/developer by helping them catch quality issues earlier and consistently deploy to production without error.

devtron

Part 3: Build a compelling business case for DX

You've done your research and background, and have designed a technical- and process-based solution to an ongoing problem you can solve with DX. **Now is not the time for wild assumptions, guesses, or fuzzy math, and your ambitions should go beyond building a bullet-pointed PowerPoint presentation you read off verbatim to your leadership.**

How you design and format your business case is up to you. First and foremost, you're aiming for a story that compels your leadership to commit to change because you've effectively illustrated how the business drivers they care about most will be positively impacted by this vague-sounding idea of developer experience. You'll also need to back up your claims with data, analysis, and carefully-crafted estimations —while you tell your story, make sure you weave in these four key elements:

Lead with your hook

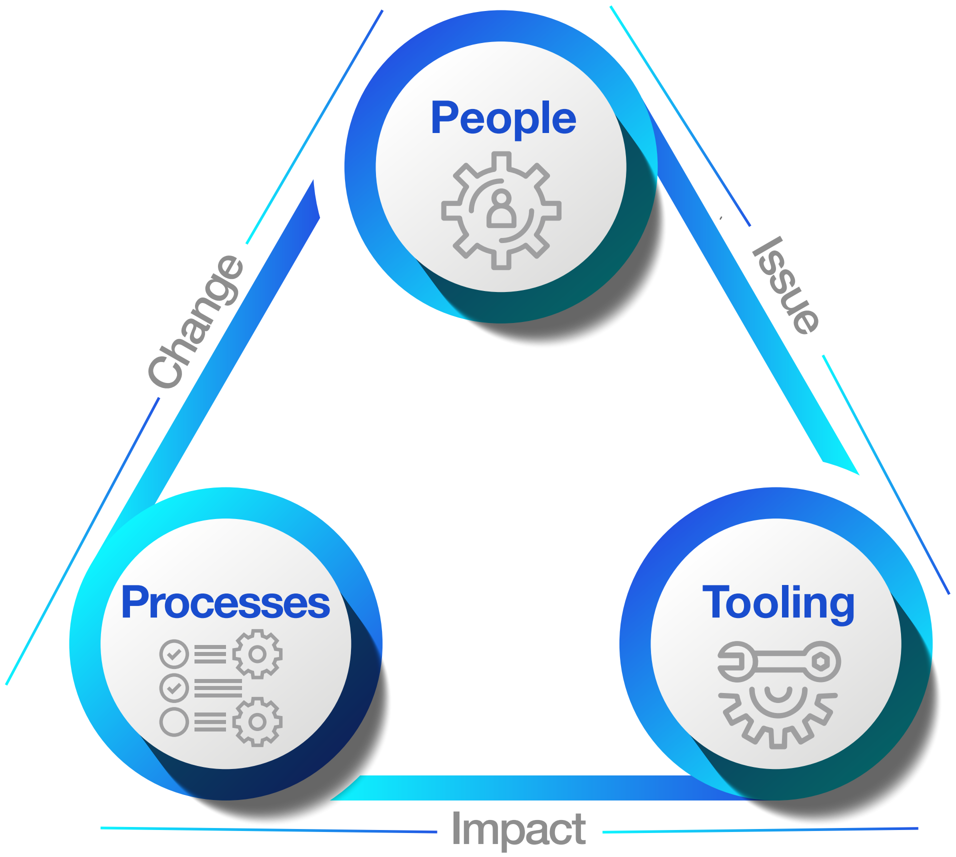
Remember that you're trying to reframe this fuzzy concept of developer experience into a tangible and immediately understandable business value. You've already created the framework of your hook, but now is the time to sharpen it down into a sentence or two that clarifies the problem, your solution, and the positive impact of investing.

If you're struggling, lean on your team, peers, and network to gather feedback and refine your messaging. You can even develop new options using a copywriting and marketing framework like AIDA to attract attention to a problem and heighten their desire for a solution.

Cover off people, processes, and tooling.

As part of your situational assessment, you should have described exactly how you propose achieving the goal. This is your opportunity to address who will drive this project, what they'll implement, and which established industry best practices you'll follow. In our companion eBook on DX, we laid out several proven DX strategies, like improving documentation, encouraging regular feedback, adding deeper observability, and encouraging developer self-service.

Like a powerful resume, specificity matters here. Instead of saying that you're going to "improve your CI/CD pipeline to remove developer downtime," specify exactly where the downtime exists, why that's an issue, and what changes or new tooling will smooth out those gaps.

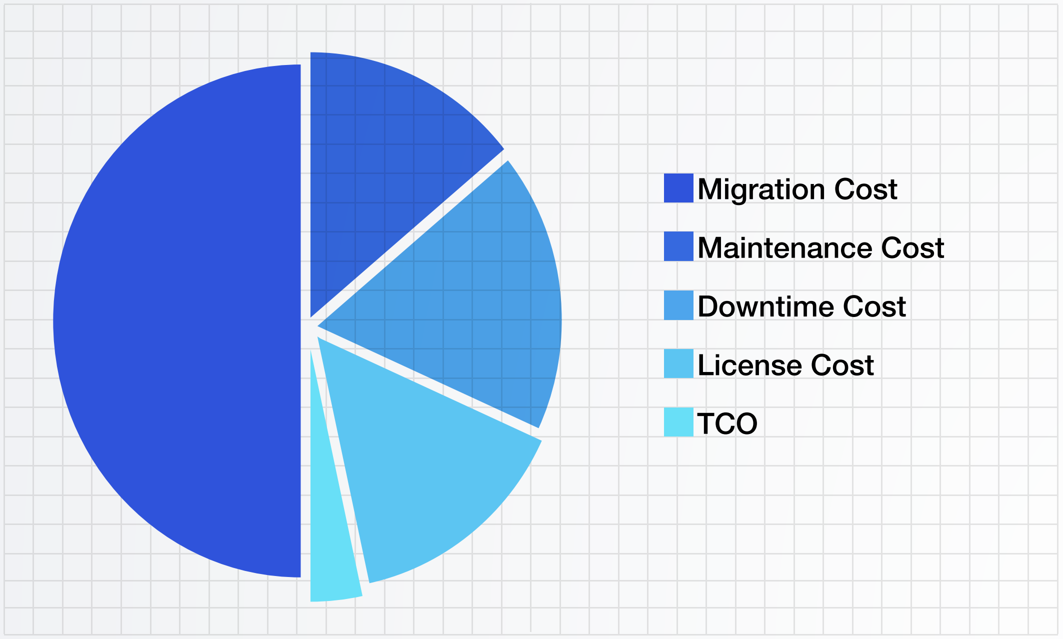


devtron

Break down all your costs.

Begin with the initial cost of new platforms, licenses, and additional cloud expenditure, along with the number of engineers and hours/days/weeks they'll devote to this project. A strong business plan includes a specific estimation, not a fuzzy estimate—cost estimates can always be revised, but a successful business case hinges on confidence. Consider using the knowledge provided by your tooling vendor, as they can provide good estimates of how long implementation takes and how many resources are required.

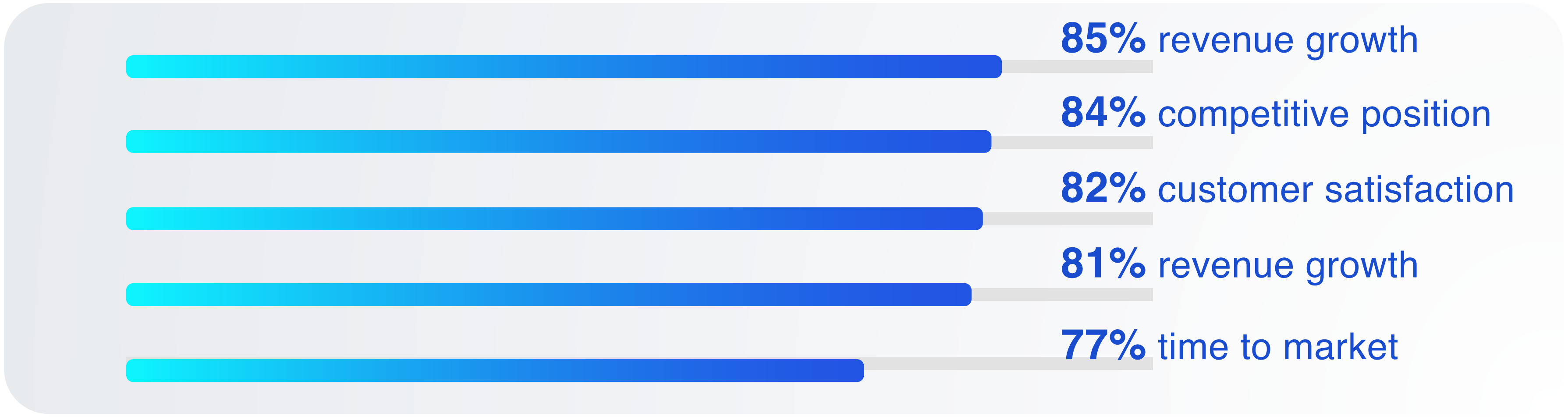
You should also tally up your plan's total cost including migration, long-term maintenance, possible downtime, ongoing license costs, and anything else that adds up beyond your proposed start and end dates.



End the cost conversation with a section on the cost of doing nothing. You should already have data and strong opinions here, gathered in Part 2 of this process. Including it here ensures your leadership understands that while any major investment comes with risk, there are also specific risks in maintaining the status quo.

Showcase the potential.

Time to drive the impact of making a proactive investment in DX home. Bubble up all your impacts into a “holistic” ROI that resonates with your leadership audience—while they certainly wouldn’t turn away taking time or minimizing errors deployed to production, they will be much more interested in how those efficiencies enable your organization at large. If you’re still looking for ways to frame the conversation around C-suite priorities, Forrester found that DX investment led to moderate-to-significant business improvements in the following areas:



Let’s return to our example DX initiative one last time to illustrate what you should aim for. You estimate that your “golden path” saves, on average, 45 minutes/day/developer, which translates to 75 wasted development hours per week across your team of 20 developers.

For an investment of two DevOps engineers and a business quarter, which translates to roughly 1,000 engineer hours, investing in DX “pays off” the time spent in roughly 13 weeks. Even when you add $2,500/month in new tooling costs, the compounding effect of saving 300 hours in developer time every month results in net savings for your organization. With those extra hours, developers can also reduce time to market by 10%, which results in healthy revenue growth and better positioning against the competition—all results your leadership audience will be more than happy to green-light.

Conclusion

Developer experience is a powerful solution, one fully validated by other organization's investments, to some of the biggest inefficiencies in your SDLC. You now have a process for building a business case around it, from identifying problems starting the right conversations with your developer customers/stakeholders, and gathering data that illustrates your status quo and gives you clear goals to propose and work toward.

You have two paths forward:

If you're ambitious and prefer to go it alone, we wish you the best of luck! Hopefully we've provided a great framework for you to follow as you build out your business case.

If you know DX should be a priority but can't quite find the right angles, let the folks at Devtron help. We've worked on DX from every possible angle, so we have all the resources and calculators to help you figure out your opportunity costs and possible return on investment. We'll work with you to develop a business case around our Kubernetes software delivery platform and the successes our customers have already achieved.

Whichever you choose, remember that while any investment in DX requires a positive return, it's also a path toward better collaboration, deeper empathy, and more tight-knit collaboration for the long haul. These impacts might not appear on the bottom line, but they are necessary for the long-term health of your engineering and development teams. When you break down one silo, you find a new opportunity for the next stage of your continuous improvement.