SPRING 2021

Transform your CI/CD Pipeline

Common pitfalls and how to avoid them

Adaptavist

Transform your CI/CD Pipeline

Common pitfalls and how to avoid them

Contents

Introduction

A framework to facilitate DevOps 3

Part One

What is CI/CD's role within DevOps? 4

How does CI/CD work? 5

The key stages of a CI/CD pipeline 6

Continuous delivery or continuous deployment? 7

What benefits can CI/CD bring? 7

Part Two

People 9

People pitfalls and how to avoid them 10

Part Three

Technology 12

Tools fit for four key phases 13

Technology pitfalls and how to avoid them 14

Part Four

Processes 17

Process pitfalls and how to avoid them 18

Conclusion

Automate, automate, and automate some more 21

Introduction

A framework to facilitate DevOps

To survive, let alone thrive, organisations are having to reorganise their IT approach, finding new ways to deliver value to the business and, ultimately, their customers. DevOps does just that – it merges siloed development staff and their processes with operational teams, enabling them to work together with a unified focus. Their objectives are aligned; their modus operandi is fast development, fast deployment, and fast iterations and upgrades.

DevOps represents a management concept – the overarching thinking that drives culture, technology, and processes to work in this way. The technical implementation framework it relies on is continuous integration/continuous delivery (CI/CD). The main goal of CI/CD is clear: customer satisfaction. And the way this is achieved can be summed up simply: by delivering products and value faster.

In this ebook, we'll take a closer look at CI/CD and its significant role within DevOps. We'll explore best practice in relation to people, technology, and processes; highlight the changes you might need to implement and the reasons behind them; and take a look at some of the common pitfalls you'll want to avoid as part of your own transformation.

Part One

What is CI/CD's role within DevOps?

As technology has advanced, so too have the processes software companies use to get their products out to customers. Agile principles and practices were born out of a growing demand for rapidity. They reduce the amount of time required to develop new functionality, encouraging dev teams to release early and often to deliver more value more quickly to the customer. But prior to the introduction of cloud-based operations and DevOps principles, deploying to production could still take weeks or months. These have sped up this process exponentially, reducing that time to a matter of hours or minutes. CI/CD is at the heart of that fast-paced sea change.

DevOps incorporates how teams utilise people, processes, and technologies to deliver value to customers, but it isn't just about getting deliverables released. For DevOps to be successful, teams need to ensure they're delivering value quickly, securely, and repeatedly. CI/CD helps teams to focus on making rapid improvements to the customer experience. It builds automation, governance, longevity, and security into software development. It speeds up the life cycle by combining build, configure, test, and deployment functions into product-oriented teams.

How does CI/CD work?

CI/CD thinking and practices are supported by tools that enable automation and regular and rigorous testing at each stage of the software pipeline. This reduces the time required to integrate changes before moving code into production. It depends on closed feedback loops to identify and fix problems fast. This notion of making ongoing improvements – small changes more frequently – makes testing more manageable and integration less problematic.

In a CI/CD pipeline, all developers merge their code changes in a central repository on a regular basis – in top-performing organisations, multiple times a day is not uncommon. Each code change initiates automated build-and-test sequences, providing feedback to the developers so they can fix any issues. Deployment might be manual with multiple stages built in, but automation is central to this part of the pipeline too. Without automation, engineers would need to perform these same steps manually, which takes a lot longer and is potentially more unreliable.

The key stages of a CI/CD pipeline

Source: a change in code in the repository (or automatic scheduling) triggers a notification to run automations in the pipeline.

Build: the source code is combined to build an artifact, a usable instance of the product, flagging any potential problems.

Test: automated tests are run to validate the code and how the product behaves. These tests might be run in multiple stages, and provide feedback to the developers.

Deploy: when code has passed the tests, it's ready to be deployed into a new environment, either manually into a staging environment first or automatically into a production environment for the customers.

Monitor: once the artifact is deployed into production, the application is continuously monitored to analyse trends, examine the performance, and proactively identify problem areas.

Any failures will trigger notifications to let the relevant developer know the cause, and when code is successfully deployed to production, the whole team is made aware too.

Continuous delivery or continuous deployment?

The CD in CI/CD is sometimes mistakenly referred to as continuous deployment. Just to clarify, *continuous delivery* refers to the discipline of building software in a way that means it can be deployed to production at any time. *Continuous deployment*, on the other hand, refers to every change being automatically put into production, so multiple deployments every day. In order to achieve continuous deployment, you must be doing continuous delivery.

What benefits can CI/CD bring?

Poor or non-existent CI/CD means engineers spend all their time on maintenance rather than revenue-generating projects, code is wasted waiting for someone to test it, money is lost through poor productivity and reliability, and developer retention will suffer. But get your CI/CD strategy right and you stand to benefit significantly.

Create code that makes you money – rather than sitting in a queue waiting for manual testing, your code gets tested automatically and deployed to production. That means you can start making money from it right away.

Increased code quality – because your people are focused on what they do best, you can pretty much bid farewell to bad code. Developers can spend more time creating good code rather than worrying about production, and ops can stop having to feel like it's holding up the process.

More innovative, more competitive – getting features to market fast helps you stand out from the competition and set the agenda for your sector, driving innovation internally.

Hire the best and keep them too – engineers want to focus on their speciality, and with CI/CD they can experience greater productivity, autonomy, and work on more enjoyable tasks. Word will get around, and you'll soon be attracting the best talent.

These benefits can be transformational, but they won't come immediately. Successful implementation takes consideration, organisation, and time. The first step is understanding best practice when it comes to people, technology, and processes, arming yourself with the knowledge you need to avoid common pitfalls. Next, we'll take a look at how you can improve your CI/CD service across these three key areas.

Part Two

People

As your organisation's primary resource, people have a big role to play in the success of your CI/CD strategy. Having the right people in place is imperative if you're going to make the shift away from a more traditional software development model to a DevOps approach. Long gone are the days of spending months or even years planning, developing, testing, and finally deploying new products, features, or upgrades. As the new mindset on the block, DevOps requires developers to focus on making ongoing improvements in smaller increments and responding to customer demands.

While CI/CD heavily impacts the way IT works, it has wider cultural implications on the business at large. It depends on a completely different approach to how and when value is delivered to the customer. Teams practising CI/CD will detect errors earlier in the development process, reduce integration problems, and develop faster and with more confidence. As a result, they'll feel more productive and happier at work. This uptick will ensure you retain the best people and attract new bright sparks to the business.

The most obvious and important change that you'll need to make is ensuring collaboration and communication among teams takes precedence and that the right technology and practices are in place to facilitate this. For CI/CD to be successful, people from different parts of the business will need to be comfortable teaming up and sharing knowledge. It's all about working towards a common goal and achieving more together.

People pitfalls and how to avoid them

Prioritising technical expertise over soft skills

People skills are very important in a DevOps team. Developers aren't confined to working in silos on individual projects – they're part of a wider group of individuals that need to communicate clearly for speedy CI/CD to take place. They should also have a customer-first mindset to help empathise with the end user's needs and deliver accordingly. Managers tasked with putting together a DevOps team should include a mix of process, functional, technical, and soft skills to find the right balance.

Assuming agile will come naturally

Terms like DevOps and agile can get bandied around, but if your team have been in place for a while, a lot of what you're introducing might be alien to them. It's important to make sure teams include a certain level of CI/CD and agile development experience to help set your strategy in motion. That means an appreciation for software application architecture too, not just coding knowledge, which will help them know what's worth automating and how.

Relying on rigid, static ways of working

DevOps demands that people are under pressure to juggle a number of projects at the same time, so multitaskers are a must. And because requirements can change pretty quickly, a more fluid mindset is essential. Shunning rigidity and embracing flexibility will mean teams are more prepared to put out fires when problems do arise.

Scrimping on security knowledge

Because CI/CD lets you develop and deliver new features faster, it opens your software up to the risk of vulnerabilities. Rather than seeing safety as an afterthought, embrace a shift-left approach, fixing security issues early and often with quality testing. This includes incorporating role-based access control and leveraging secrets management to ensure sensitive information is properly protected. Your teams will need a solid understanding of best practice for building secure software to keep data safe and minimise the threat of cyber attacks.

Overlooking automation and analysis

There has been a proliferation of DevOps technology in recent years, so good team knowledge of the right automation and testing tools is key, with a focus on technical cloud skills and data set analysis. Lots of these tools are open source and more are being introduced all the time. To scale and grow your own DevOps environment, encourage your people to stay on top of the latest innovations.

Part Three

Technology

The most important thing to remember when assessing your CI/CD tooling needs is that it's not a case of one size fits all. Where you're at now and where you want to be will be different for every organisation, so the first step is to understand your current application mix.

Are you still dependent on monolithic applications or is your environment based on a microservices architecture with a container-based pipeline? The latter is where most organisations are heading, enjoying the benefits of decoupled software elements, reusable components, and independent development cycles. Chances are, you're not there yet or you're somewhere in between, and this will play into your tooling choices, enabling you to measure the impact technology has on your environment.

Another key consideration is what you can afford. While many tools are open source – giving you access to a thriving community to solve problems – you'll need to factor in support costs and hosting services. Paid software can bring its own advantages, such as expertise for your configuration at your fingertips. You should consider which tools will help you produce better quality code, reduce vulnerabilities, and increase operational efficiency.

Tools fit for four key phases

There are some key phases to any CI/CD pipeline, namely source, build, test, and deploy, and you'll need tools to support each. While we're not recommending any specific tools in this ebook, here are a few factors to consider for each phase:

Source – pipelines are typically triggered by a source code repository. A version control system that tracks these changes, notifies the CI/CD tool, and runs the relevant pipeline is a huge benefit. It will let you manage multiple programmers working across the project and helps to speed things up.

Build – this is when the source code and its dependencies are combined to build a runnable instance. Here container software can be used in cloud-native environments. When done right, it allows you to speed up building, testing, and deploying software, features, and functionality. And it lets you do all this more easily and frequently without disrupting the user experience.

Test – automated tests validate the code and bring any issues to light. Larger projects require lots of testing of different types at several stages. Tools that can automatically build, document, integrate, test, facilitate required changes, and prepare an application for deployment help keep this phase on track.

Deploy – DevOps teams will largely deploy work-in-progress to a staging environment for additional testing and review. You'll want tools that can automatically deploy from the main or 'release' branch to the production environment for your end users.

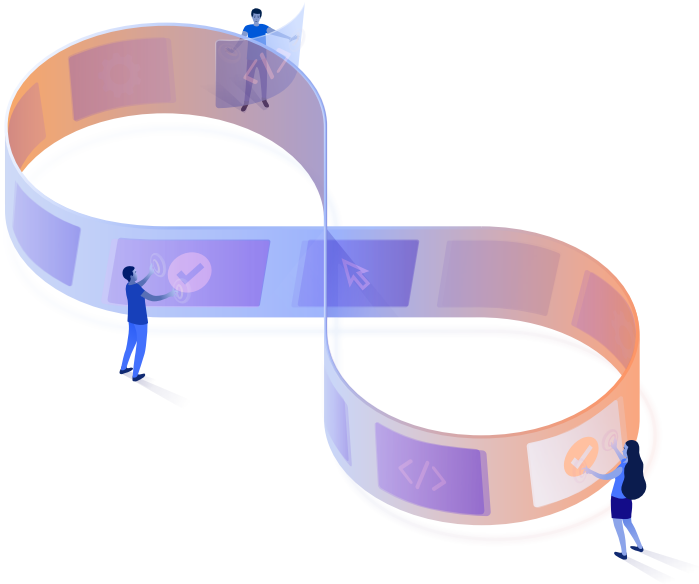
Technology pitfalls and how to avoid them

Not considering customisation needs

How much customisation are you able (or willing) to do on your own? What you decide will have an impact on costs, as well as how long it will take to implement your new tools. You'll also have to consider the resources you need going forward in-house or whether you'll pursue other avenues, such as consultants, which could be pricey. Knowing what you can afford will play into what you choose, but don't overlook the importance of tailored tools. You need technology that works for your organisation, and customisation is a part of this.

Choosing disparate tools with zero integration

Related to customisation is integration. A big chunk of your overall IT budget goes on supporting teams as they maintain and integrate a complex toolchain. The easier you can make this, the better. Don't fall into the trap of picking tools with no integration features. You need to consider integration from two angles: how integrated is each tool for its specific function and how easily does it integrate with the entire pipeline environment?



Being unable to put the work in to use a tool

Signing up for something and actually incorporating it into your pipeline are two separate things. It's easy to get excited about the potential benefits of a tool without considering the work involved in getting everyone to use it. Just because something requires effort and change, it doesn't mean it's not worth doing, but make sure you're aware of the challenges ahead so you can plan for implementation.

Ignoring larger environmental needs

The same goes for doing your homework as far as your larger environmental needs are concerned. As mentioned above, think about where your architecture sits on the scale of monolithic to microservices, what your workflows are like, and the skillsets of your people. Don't ignore questions around hosting – does your provider already offer CI/CD technologies? Where is the data your applications rely on going to be stored? Will the tool you're considering be able to handle new linkages and maintain the performance you need?

Having unreasonable automation expectations

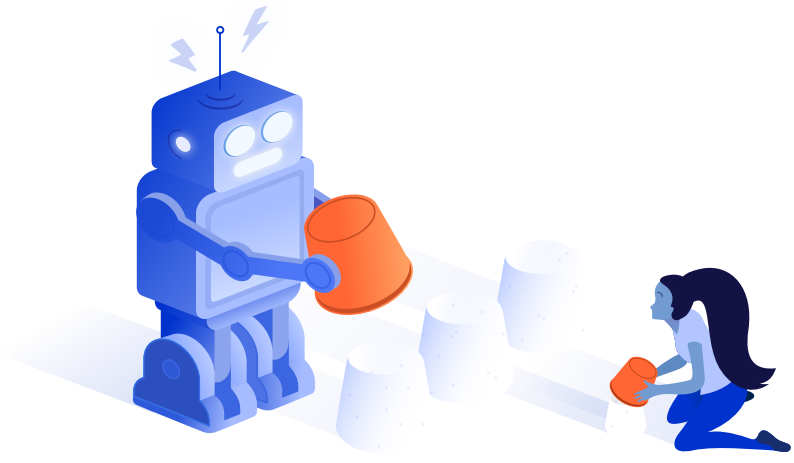
In theory, successful CI/CD means automating as much as possible, but you need to understand what that looks like in reality. Organisations with a mature DevOps culture are capable of deploying code to production hundreds of times per day, but zero intervention between development and customer interaction isn't optimal for many applications. The goal of one-button automation should be measured against the importance of human checks, to ensure compliance, for example. You want tools that enable and encourage automation but allow for manual reviews too.

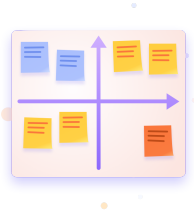
Sidelining security tools

Don’t forget about security – you need to account for tools that allow faster issue identification, notification, and better visibility so you have a clear idea of the risks and vulnerabilities at each stage of the pipeline. But an overly complex toolchain could weaken security. You might need to pay more for tools with additional security functionality, but these, combined with an efficient CI/CD process, will ensure data is safe, risk is low, and there is efficient issue resolution when problems do arise.

Forgetting about the future

Don’t get too caught up in the here and now. When evaluating tools, think about what comes next. Accommodating growth, headcount goals, expansion plans, additional products and services, and M&A activity into the new applications you choose today will avoid having to retool down the line. Ask yourself whether free software will give you the room you need to grow or could it limit your plans and your ability to hit the numbers you're projecting?





Part Four

Processes

With your people prepared and tools in place, it's time to turn your attention to the processes that facilitate a fast workflow with CI/CD. Keep in mind that your processes should work *with* your people to help them deliver value to customers, rather than making lives more difficult. With powerful processes, integration becomes a non-event and new products and features can be rolled out right when customers need them.

The first step is plan. The second step? Plan some more. CI/CD processes require a healthy dose of up-front organisation. Lots of the processes you're going to implement will be automated. Relying on automation can't be rushed. You need to establish a solid framework that's tailored to your organisation and business needs. What that definitely *doesn't* look like is leaving teams to figure things out for themselves, hoping the best processes will surface during execution. Far from it.

Sure, things will change and need to be adjusted as you move from the planning phase into execution. You'll go from focusing on big-picture considerations to sweating the small stuff, looking at the details and improving where you can. It's IT management's job to hold everyone accountable for sticking to the plan or to justify changing it.

Process pitfalls and how to avoid them

Planning for success

As with people and tools, establishing secure processes is vital to the future of your pipeline. Rather than planning for a perfect world where security isn't a concern, you need to plan for failure, attacks, and other risks. Include security teams in your process planning at the start. Containerisation is one strategy that can help, allowing internal application processes to be isolated from one another. Automating testing is important and should be a priority alongside compliance testing. Consider inserting security gates into the pipeline too. That way human intervention can confirm all protocols have been followed prior to release.

Running before you can walk

It might sound obvious, but going from zero to a hundred can cause problems. Lots of organisations choose to get started with CI instead, building out their processes, before incorporating CD as well. Remember the distinction we made between continuous delivery and continuous deployment in part one? It's worth remembering here that focusing on getting good at CD is about building operational and management confidence. Focus on that first, and then you can worry about automating deployment too.

Not considering consistency

Getting to a place where you're comfortable with your CI/CD pipeline requires you to establish a consistent cadence of builds and deployments, helping you understand the pace your teams work at. You'll want to have clear rules around what triggers an automated build or deployment. Version control (a single source of truth for all teams with artifacts in one repository) and issue tracking are vital here. Don't worry if what you plan for might not be what's needed, just be prepared to adjust with experience.

Being noncommittal

Early and often is the way to go when it comes to integrating code into the main branch. Working this way will avoid maintenance nightmares for your engineers. For work in progress, code can remain invisible to the end user or tester. Progressive delivery processed, like feature flag management, will help to navigate and minimise merge conflicts.

Ignoring different types of deployments

While deploying often is the goal, don't forget that not all deployments are created equal. There are different options depending on business needs, so you might want to include three frameworks, such as canary, blue/green, and A–B testing. The first is a minimum viability product type release for fast iteration; blue/green tests internal processes, such as different routing methodologies, to help you figure out what works better; and A–B testing is fairly common and customer-facing. It reveals clear preferences in conversion, which can be spread elsewhere in the application.

Only testing code

While it's essential to test your deployment pipelines to the max, mitigating defects and making improvements, don't forget to put your people to the test too. Part of quality management is making sure your developers are checking into the trunk at least once a day, that every check-in triggers an automated build and testing, and that if a build breaks or a test fails, the problem can be solved in a few minutes.

Not making enough of metrics

Establishing and tracking metrics should be a big part of your initial processes discussion. Some typical data points you should try to measure include quality, speed of delivery, deployment frequency, change lead time, change success/failure rate, mean time to recovery from pipeline failure, and passing rate of security tests. Monitoring your chosen metrics over time lets you spot emerging bottlenecks as things evolve.

Conclusion

Automate, automate, and automate some more

At its most basic level, your CI/CD strategy should be based around automation – processes that support automation and keep it in check, tools that enable automation to thrive, and people with the skills to plan and implement automation while focusing on more important work. Anything that can be repeated, such as building, testing, and deployment, is ripe for automation. And anything that's automated should benefit from human intervention where it's prudent to do so.

CI/CD thinking, technology, and processes are what will enable your organisation to keep up with the fast-paced global economy. They'll set your software apart from competitors', giving you an advantage in a crowded marketplace. And they allow you to keep evolving, improving the quality of your software for your customers. Internally, your teams will also benefit from increased efficiency and be able to channel their skills into so much more than mundane, repetitive tasks.

CI/CD is an intrinsic part of making your DevOps approach a reality. It's not something to be taken lightly or rushed into, but with careful planning and a bit of courage, you can take steps towards a transformative future for your organisation.

At Adaptavist, our DevOps solutions help you stay ahead of the game. With our strong and dedicated team of experts, we combine the right mix of strategic-led consultancy and technology-led solutions that place people, process, and tools at the heart of your business strategy.

To discover DevOps in a whole new light

Contact us

Adaptavist

We help organisations transform to continuous change being their business as usual. We do this by supplying technology, providing advice, and delivering change through modern, iterative approaches to development, deployment, and application lifecycle management.

Adaptavist is Atlassian's largest platinum partner, supporting more than half of the Fortune 500. We are uniquely placed to provide our experience, expertise, and insight to help your business.

Whether you want training for your team, to build a software platform for your company, or to automate your existing tooling, we can help you. If you want to unlock the full power of Atlassian and transform your business at scale, get in touch with our team today.

Learn more or get in touch:

adaptavist.com

ATLASSIAN Training Partner

ATLASSIAN Platinum Solution Partner ENTERPRISE

ATLASSIAN Platinum Top Vendor