PARALLO
a rhipe company

Greasing the SaaS release pipeline

Proven DevOps enablement strategies for ISVs to optimise CI/CD pipelines in Microsoft Azure

October 2020

by James Brookbanks , Azure Service Manager

A PARALLO EBOOK

Microsoft Partner
Microsoft
Gold Cloud Platform
Gold Application Developm
Gold DevOps
Silver Security

Azure Specialist for ISV

Contents

Chapter 1

Is friction causing inefficiency in your SaaS application delivery process?

Developing a brilliant solution to a customer need is at the core of any great software solution, but a highly profitable one removes the friction from the process of getting that solution to paying customers. This eBook is aimed at helping ISVs better implement DevOps to streamline their CI/CD pipelines and speed up release cycles.

Playing catch-up is inefficient

Periodic approaches to product deployment, such as quarterly release cycles, create huge pressure on internal development teams while impacting negatively on customer experience. The build-up of fixes and feature improvements over a quarter or an even longer period of time require slow and careful releases to customers, who require a pre-production environment for regression and user acceptance testing before a new release goes live. The complexity of these large releases inevitably result in new bugs and slow the process further.

In mission-critical environments particularly this periodic release process can become glacial, with quality control sometimes meaning releases only go into production right before the next release arrives. ISVs and their customers become stuck in this fundamentally inefficient mode.

At best, the periodic approach slows down the process of delivering the benefits of product improvements to customers, at worst it results in downtime. For customers who expect technology to be always-on, this causes frustration.

The continuous deployment method inherent to a DevOps approach 'greases' the release process. Continual 'micro-releases' and UAT mean bugs are found earlier and addressed quicker, and new features get into the hands of users faster.

Deployment and application maintenance costs are reduced, while customers' satisfaction increases. So how can you move to this agile model of software deployment and better manage your release pipelines?

Chapter 2

A DevOps approach to enable rapid, high-performing releases

SaaS can be a highly profitable area of cloud business because it promises low to no administrative burden, with much of the heavy lifting performed by a service provider.

But to really get the most out of your product and achieve significant growth and returns, technical fine-tuning is required. Many ISVs may not be able to commit the time and resources this requires, however delivering features quickly and reliably, enhancing CX, and ensuring a resilient, scalable application environment is just the tip of the iceberg when it comes to the positive outcomes of a successful DevOps roadmap.

Correctly integrating DevOps throughout the lifecycle of your SaaS offer is a proven approach to avoiding operational difficulties for both grassroots SaaS entrepreneurs and established enterprises.

So what is DevOps at its core?

It's the bringing together of the development and operations teams. Historically, development teams would build a software product, then 'throw it over the fence' to operations for installation on a server and ongoing management. Inevitably, this could be the cause of tension between the teams, especially if operations weren't sure what the software was intended for. Lack of communication between the teams also meant blow-outs on project time frames.

Combining these disciplines helped them to be on the same page from the beginning, especially with regards to the production environment.

A 2019 State of DevOps report found that businesses who decided to implement DevOps can expect:

- Faster delivery velocity
- Higher flexibility for development dealing with different applications, services or components
- Fewer security issues
- Lower change failure rate
- Better reporting of metrics that can be shared across the business thanks to automated systems

Additionally, the DevOps approach means that operations have more influence on how software products are built, so that the areas of security, performance and availability are effectively 'baked in' to the product.

Architecting your DevOps process

It's important for ISVs to have a clear idea of what they want to achieve. Is it minimal downtime? Or a faster release of features? ISVs need to identify these measurable business outcomes as they are the foundations of DevOps transformations. The approach drives a new set of practices, culture changes, and ways for ISV teams to collaborate on development and operational goals.

Moving to 'cloud cadence' – which means changing schedules and developing a rhythmic culture or routine - utilises the concept of one 'engineering' system - a single place for version control, reporting, requirements management, project management, automated builds, testing and release management capabilities. This creates a unified build rather than having separate roles, meaning that everyone on the team has a greater impact on the quality of the software.

Chapter 3

Implementing scalable CI/CD pipelines to reduce friction in release management

The aim of your software application is to get the end product to your customer faster than ever. To do this, while minimising the risks involved with each build requires an accurate CI/CD pipeline setup.

These pipelines are industry best practice for DevOps teams, enabling code changes to be delivered more frequently and reliably.

What are they and how do they help?

Continuous Integration

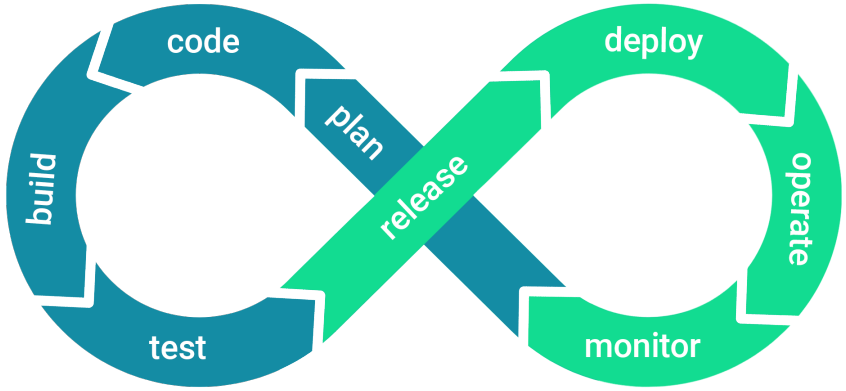
The process where the entire developer team 'commit' their code to integrate it into a shared repository – sometimes multiple times in a day. Compared to the traditional approach of integrating separate instances of their individual code into the main area.

To minimise errors, CI involves a range of testing to ensure a successful build. The idea is to establish a consistent and automated way to build, package, and test applications.

Continuous Deployment and Continuous Delivery

CD automates the delivery of applications to selected infrastructure environments. Once the CI has succeeded, CD picks up the software release and pushes it out on to the production infrastructure, where it runs and is available to end users.

Continuous testing is implemented as a set of automated regression, performance, and other tests that are executed in the CI/CD pipeline. The DevOps approach ensures that any code is not deemed complete until there has been an automated test written for it.



Chapter 4

Release management - the good, bad and the must-haves

In the past, release cycles have been quite long and spread out involving weeks or months of tedious planning, development, followed by testing and even a stabilisation period. With CI/CD pipelines, code is constantly built and deployed with little to no time between cycles. This results in a higher number of frequent releases.

ISV's today need to focus on shortening these release cycles by employing proven DevOps release management strategies, one of which we highly recommend and explore further below:

The 'Release Flow' method

The strategy based on Git and Azure DevOps, is a way to use version control and branching to deliver changes safely to production, allowing developers to rapidly build, test and deploy frequently.

Trunk-based branching strategy is a tried-and-tested approach to version control and branching to deliver changes safely to production. The Release Flow method consists of 4 steps, from development to deployment:

1. **Branch -** The first step in this process is where a developer who codes a fix or implements a feature creates a new branch off the main 'master' branch or 'trunk. It's recommended to keep these short-lived and lightweight – lasting only a couple of days at most. The branch is worked on by one or two developers (pair-programming).
2. **Push -** When the changes are ready to be integrated with the main code, they push their mini branch to a server and issue a pull request. This is done when the feature branch is ready to be closed out.
3. **Pull -** Pull requests are a way to control the quality of code entering the main master – through a series of automated tests and peer code review. The idea being that other developers objectively and speedily review new code to ensure its release-ready before it's merged into the main product.
4. **Merge -** Once everyone has signed off the code, the pull request is complete, and the feature/ fix branch is now merged into the main master. Acceptance testing gives it another layer of validation.

Releasing on shorter cycles reduces the time taken to consolidate all these branches and resolve merge conflicts, as well as fixing any dependencies that might have been broken.

Because everything is being constantly tested, it's much more stable when it's released into production. It's also worth noting that in the rare case that a problem escapes testing and causes something to break when a product has been released, the DevOps approach provides a much faster path to finding and releasing a fix for it.

Allowing for flexible changes and hot fixes

'Hotfixes' or 'software patching' is easier and faster because automated pipelines have already been set up, meaning that once a 'hotfix' has been added, it can be quickly released with minimal impact on production.

To do this, a branch is created, reviewed and merged. This allows for quick validation and seamless integration back into the master. It also ensures that the change isn't accidently left out of the master resulting in a recurring error.

Is this the right approach for you?

If your development teams find themselves manually building and juggling multiple production environments and repositories then a trunk-based approach might be the release management silver bullet you're looking for. It makes working in distributed teams easier, allowing teams to work on separate parts of their project and join their results consistently and cohesively into the main product.

It provides faster software development speed and reduces tedious processes. This makes it the perfect strategy for ISVs focussed on providing their customers with new, robust features or allowing for flexibility within the product itself (for example, pivoting due to unforeseen circumstances like a global pandemic).

Trunk-based development needs a mature build infrastructure, streamlined processes and disciplined development teams.

Chapter 5

Freeing yourself to focus on success: the case for Managed DevOps

The concept of outsourcing DevOps can enhance a company's competitive advantage by enabling faster, more agile SaaS delivery. Using DevOps through a reliable managed services provider is a perfect fit for software teams who don't have time and/or the expertise to manage a complicated application infrastructure but need it ASAP. It frees up your workforce to focus on what is strategically important for your business.

An ISV's development team typically focus on paid feature requests, product roadmap items, and customer impacting bugs with no Ops work-around. Having a managed DevOps service in place means you can keep the tech debt, which builds up in the backlog, to a minimum, while enabling the adoption of new technologies that move the needle. The outsourced service can clear out that backlog, ensuring that your app is leveraging the most efficient features, and not getting left behind.

Parallo has extensive experience providing this service across a wide range of SaaS customers. Our team of automators and orchestraters can automate environment builds, deployment pipelines, self-healing corrective actions and much more. The Parallo DevOps team rapidly fill the gap between Ops and Dev, providing value to both.

Using DevOps to drive growth

We use the Azure DevOps tools to add maturity to ISVs development teams and their processes. For many of our customers, we've developed a robust, automated CI/CD release pipeline process, meaning they can now test and deploy many times a day with confidence. This enables them to test and bring new features to market faster and more efficiently than before. It also means they're using best practice DevOps in their team, implementing small releases frequently, not 'big bundle' releases spread far apart.

We work with our customers to solve business issues with Azure DevOps, such as job lifecycle monitoring and management solutions. A solution of this nature delivers significant business impact, by providing immediate feedback and the ability to fix issues on their platform before they impact greater numbers of customers.

Because everything is being constantly tested, it's much more stable when it's released into production. It's also worth noting that in the rare case that a problem escapes testing and causes something to break when a product has been released, the DevOps approach provides a much faster path to finding and releasing a fix for it.

Eliminating friction: from 50% failure rate to 0%

One of our customers, a New Zealand ISV, had two key challenges around product development:

1. Product testing and release process
2. No solution for the monitoring of in-process jobs

They used three environments across their application. Each environment required at least a 45-minute build time before new features or bug fixes could be tested or applied.

Over time, the failure rate for the build process rose to more than 50%, despite the development team spending significant time hand-holding each build cycle. It got so bad that the team began testing on the production environment.

Parallo re-architected all three environments and built a new automated build and release process using Azure DevOps, resulting in a complete build time across all three environments in 5 minutes, with 0% failure. The company is now highly agile and responsive.

Can you afford not to reduce release friction?

Friction in the product release process creates inefficiencies that are costly for ISVs, while impacting on the satisfaction of your paying customers. Successful SaaS enterprises are those with not just great IP, but smooth and continual releases.

CI/CD pipelines are the ideal approach for ISVs that want to improve applications frequently and require a reliable delivery process. Its aim is to standardise builds, develop tests, and automate deployments; in a nutshell, it's the manufacturing process for deploying code changes.

Once in place, DevOps teams can focus on the process of enhancing applications and less on the system details of delivering it to production environments.

It is time to get out the grease gun for your product release pipeline.

If you'd like to find out more about how DevOps can keep you competitive, and be harnessed for business success, don't hesitate to get in touch and we'll get a discussion started.