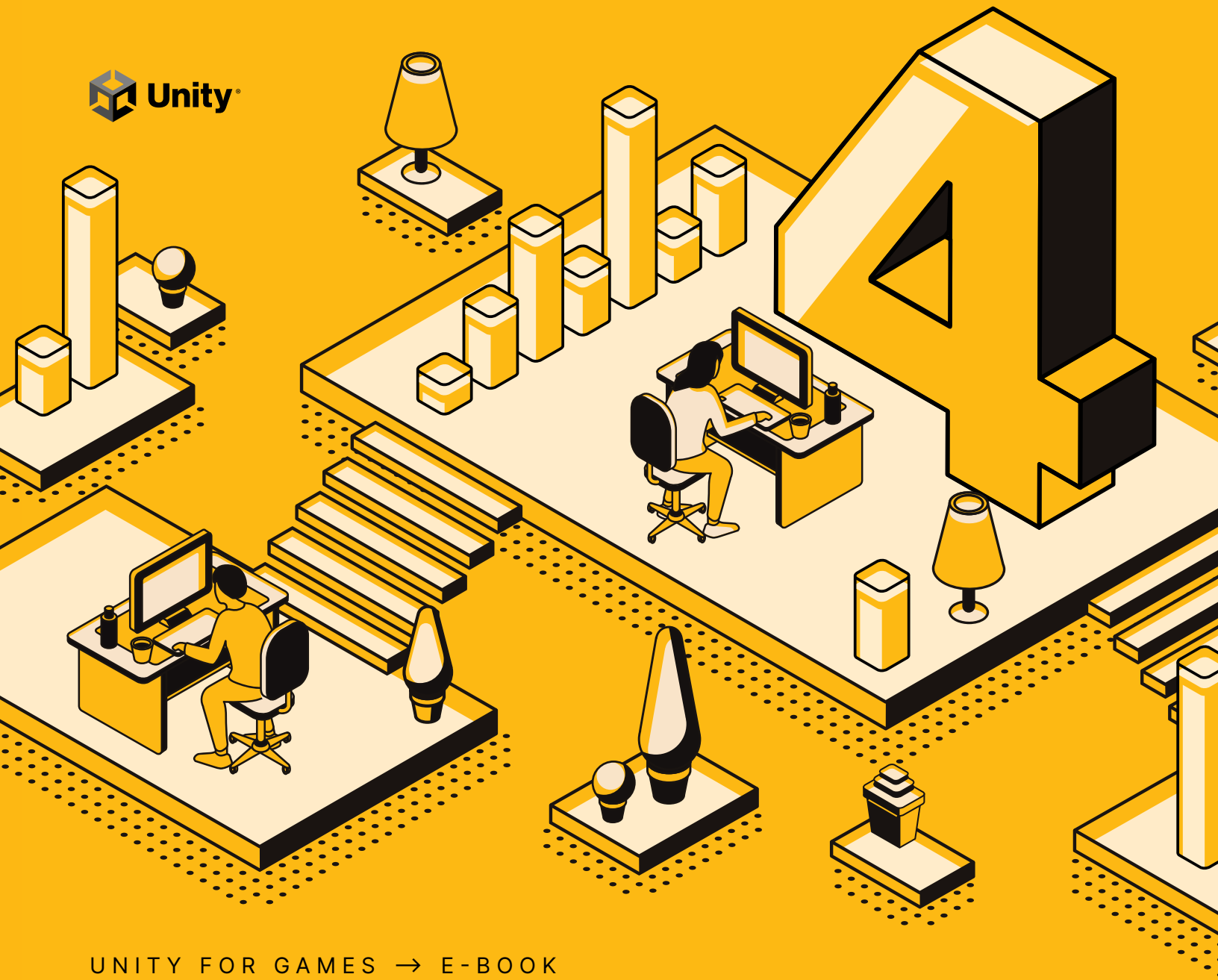


UNITY FOR GAMES → E-BOOK

FOUR ESSENTIAL DEVOPS PRACTICES TO MAKE ALL YOUR GAMES STAND OUT

Move from hot-fix crunches to rapid iterations and releases.

INTRODUCTION

Working 60-hour weeks, months before a release? Slinging code all night to deliver miracle bug fixes several times a week? Adding on last-minute, inspired features literally right up to the launch?

This industry is notorious for its crunch-crunch-crunch cadence. And you may be someone who thrives on that craziness. But if the payoff isn't there at the end of the fiscal year, you've got to be asking yourself, *why*?

Maybe a launch crunch was too much for one of your key developers, who quit and delayed a release. Maybe some all-night fix kicked off a chain reaction of follow-up all-night fixes. Maybe a last-minute, inadequately tested feature threatened to bring the whole project crashing down. Whatever the specific scenario, one thing is clear: You've been here too many times.

If these tactics don't feel worth the price, it's because despite your heroics, you're not giving customers what they want. The ability to script shaders, animate water, imagine levels, or configure a CDN are only part of what it takes to successfully create value for the gaming community. To cross the chasm into profitable production, the more crucial skills you'll need are applying systems and processes that steer your energies in one direction: delivering great games for your fans.

It takes smart DevOps practices to transform a studio from crunch-based chaos to a land of productive, effective workflows. The goal is to give your team back their most precious resource – time – so they can work on the stuff that actually matters.

But how much time can you actually get back by eliminating crunch? *Years*, according to Josh Nixdorf, a technical director at Electronic Arts:

> **“From my team’s perspective, one of the most important metrics we track is *years of effort saved*. That too has increased exponentially as we progressed on our *DevOps journey*, starting at about 20 years of effort with our first crack at DevOps and surpassing 1,000 years today.”**

— Josh Nixdorf, Technical Director at Electronic Arts

By moving to a more streamlined and intentional development process, you empower your team to deliver better experiences to your players – and you’ll gain the ability to deliver more often. This e-book introduces some fundamental DevOps principles and processes that are specific to game development environments and highlights a few essential DevOps tools for a smooth-running studio’s technology stack.

CONTENTS

WHY DEVOPS?

Game development gets more complicated with the release of each console generation. Players increasingly expect better graphics, frequent post-launch updates, and stellar support. If your game is full of bugs at launch, it's extremely difficult – if not impossible – to recover. As Xsolla president Chris Hewish said in this *GamesIndustry.biz* interview, "You can always build upon success, but you can't really grow from the ashes of a failed launch."

To maximize player value, it's essential that you examine your processes and tooling, then identify opportunities to optimize. Will your current workflows allow you to support your game over time? What about future projects?

Studio teams tend to focus their attention on how developers and artists can accomplish increasingly difficult tasks. But this focus makes it all too easy to completely miss an understanding of process. For example, if a programmer stops and checks their work more often (a DevOps process), they'll spend less time fixing bugs.

> **"YOU CAN ALWAYS BUILD UPON SUCCESS, BUT YOU CAN'T REALLY GROW FROM THE ASHES OF A FAILED LAUNCH."**
>
> — Chris Hewish, President, Xsolla

Without this understanding of process, studios are bound to endure:

- **Overworked developers who are perpetually working in crunch mode**
- **Releases plagued by bugs and negative feedback**
- **Pervasive frustration caused by confused development plans**
- **Wasted effort across the board**

The solution is to adopt DevOps practices and tools that optimize and shorten development cycles to yield consistently performant games. The benefits are too many to list, but here are a few:

- **Automation of processes that free developers from repetitive tasks**
- **Proactive rather than reactive decision-making**
- **Continuous integration and deployments that give your players higher-quality content**
- **Far less overwork, stress, and burnout**

And the payoff at the end of the fiscal year? Greater than you can imagine: happier staff, happier customers, and happier accountants.

THE FOUR ESSENTIAL PRACTICES

You don't need a DevOps engineer to start getting the benefits of a more-DevOps-like approach in your development environment. Even a solo developer will improve their productivity by simply understanding these four essential DevOps practices.

1

Robust version control: For almost any programmer, backing up code changes is as automatic as blinking. A simple commit ensures that work is safe and won't get overwritten and lost. But game development involves managing and tracking a much wider variety of sometimes-huge assets generated by distributed artists and others. This dictates a full-on version control system with a powerful feature set.

2

Build automation (CI/CD): Continuous integration (CI) means automatically merging work from one or more sources into a central repository. Continuous delivery (CD) is automatically creating new builds as you develop. Implementing an automated CI/CD pipeline is an established best practice in agile methodology, providing a consistent and automated way to build, package, and test your game.

3

Artifact management: Beyond code and assets managed by a version control system, artifacts include a wide variety of dependencies such as base images, software packages, Helm charts, metadata, and other tangential but critical resources. An artifact manager complements version control by maintaining a central repository for absolutely everything needed to build or deploy apps or games.

4

Frequent and automated testing: A DevOps approach breaks down traditional developer/QA silos. For example, developers write code in smaller chunks and perform their own tests before pushing it to the main codebase for builds. Ideally, this testing is continuous and automated. Release candidates are more refined, and QA teams dig much deeper into game performance.

You can learn more about these practices in the following pages, which dive into each of these essentials in greater detail.

— 1

ROBUST VERSION CONTROL

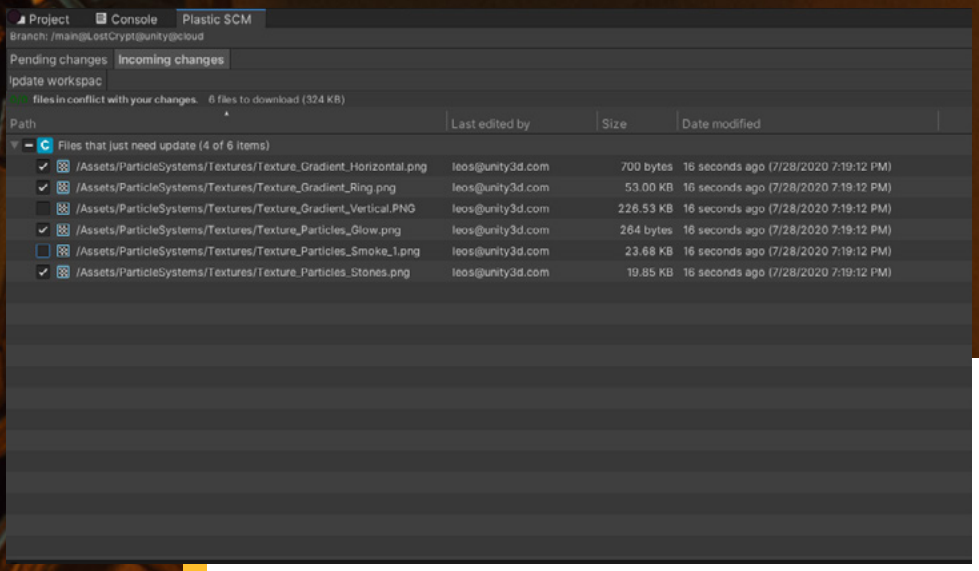
Minimize merge conflicts and lost work

When artists and programmers are constantly butting heads, or when merge conflicts result in redundant or lost work, version control is usually the culprit. To apply version control best practices in game development environments, you'll need a system with unique capabilities.

Practically speaking, code files are small, regardless of how long they are. However, graphics for next-gen console and PC games can be comparatively huge, making network bandwidth a critical factor. Pulling and pushing these assets from a central repository has to be as fast and easy as it is for any other development asset.

The biggest problem with most version control solutions? Usability – for everybody.

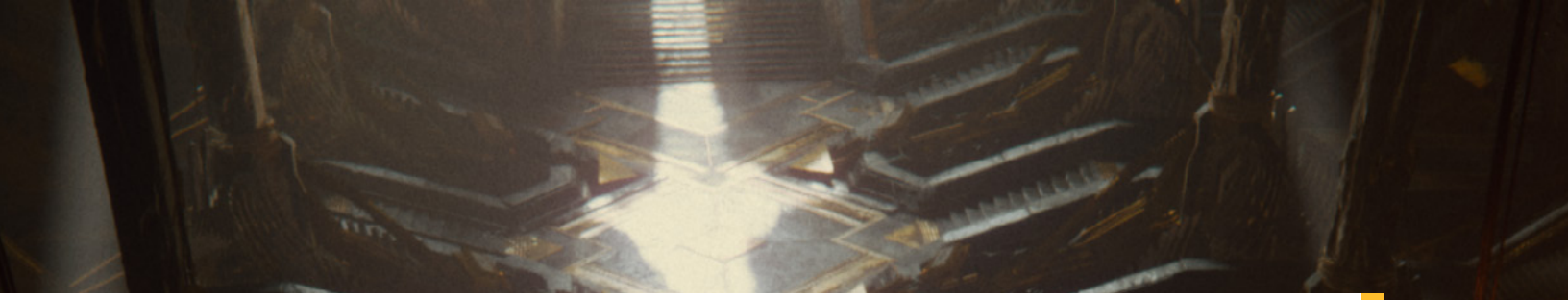
Programmers are typically adept with command-line interfaces and complex version control branching techniques, while artists may find it challenging to simply hit Save often enough. But their drawings and animations are just as critical to the final package as the codebase is.



A game development version control system has to provide complex, detailed options for developers yet also be very accessible for less-technical users. Otherwise, engineers will be spending their time explaining, over and over again, how the system works.

An intuitive, graphic UI can make complex branching and merging much easier to coordinate with the programmers, artists, and designers working concurrently on different levels and versions destined for different platforms.

Traceability means being able to see exactly when and where an asset was modified and on which branches it's used, and it's an important capability for QA teams as well. Plus, adequate security across all your repositories, branches, labels, and paths is a minimum requirement.



— 2

BUILD AUTOMATION (CI/CD)

Spend more time creating, less time managing complex pipelines and tools

A consistent theme of all DevOps practices is breaking work down into smaller chunks. Smaller units are easier to handle, simpler to test, and faster to optimize. You also minimize wasted effort if you hit a wall or decide to change direction.

In game development, that amounts to frequently merging your ongoing changes with others' and creating new builds. Manually creating builds isn't that complicated, but it's an interruption of other work, and the time it takes can add up quickly. Again, like many other DevOps practices, automation is key.

An automated build pipeline ties into your version control system, monitoring changes and creating new builds with each commit or as the number of changes pass a particular threshold. This means developers can spot and correct shortcomings early and often before things get out of hand. It keeps code and assets in a more reliable, release-ready state. If you need to roll back changes for whatever reason, the last successful build is ready to go. Plus, other teams can easily access the builds for updates and give timely feedback.

Culturally, a DevOps approach to build automation shifts your team's priorities to making sure their work is done right before they jump ahead to new features. They move forward incrementally, building on small successes, rather than doing long sprints and falling down exhausted. It builds confidence and significantly reduces the likelihood of derailing someone else's work because your code wasn't quite finished.

— 3

ARTIFACT MANAGEMENT

Get your game where it needs to go

After the CI/CD system creates a new build, it typically pushes the build to an artifact repository, where the artifact manager registers it with all its dependencies and other necessary components. Orchestrating these “other necessary components” is why artifact management is such an important practice in game development DevOps.

Artifacts can come from all kinds of sources, such as your in-house developers, contract artists, and any number of public repositories for open source code. The artifact manager is what ensures that all this critical content is uploaded and archived while checking versions and approvals, scanning for security issues, and so on. There are many possible points of failure.

Of course, any useful artifact management system has minimum requirements. It must be able to manage metadata to support versioning and filtering based on licensing. It needs to let developers set which content gets retained and for how long. And, importantly, it must be built as a high-availability system to avoid downtime.

Compared to simply designating some S3 bucket as a project repository, there are considerable benefits to implementing a real artifact manager. It means maintaining a common library for all project resources enhances collaboration, particularly between artists and engineers. All resources are registered and traceable, which makes for predictable and repeatable builds. And it frees up everyone on the team from managing deployments, so they can code, animate, design, or do whatever it is they do best.

— 4

FREQUENT AND AUTOMATED TESTING

Respond to feedback and work with agility

Two principles shape effective DevOps practices for QA:

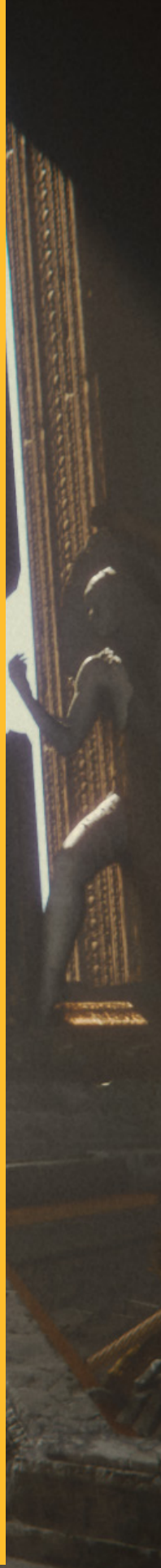
— Frequent testing early in development
— Automated QA workflows

A carpenter measures twice to cut once. In other words, they solve for the length variable and then test their work before moving on to the next production step. Developers can do the exact same thing by writing shorter code blocks and double-checking them before writing more. The line between the developer and QA blurs.

Frequent code checks by developers pay off when builds undergo automated tests. Whenever the compiler pushes a new build to the repository, it's tested. With continuous automated testing, it's much less likely that bugs will survive beyond development to cause problems in production code. And those that do survive won't be as virulent.

That said, we all know that bugs happen. What goes along with automated QA testing is automated reporting, both from testing protocols and, after a game hits the market, from the devices it's running on. As soon as QA or a user installation encounters anomalies, the system reports specific crash and exception data to enable faster fixes. Often, issues are not yet apparent to the end user and are fixable before surfacing as noticeable glitches.

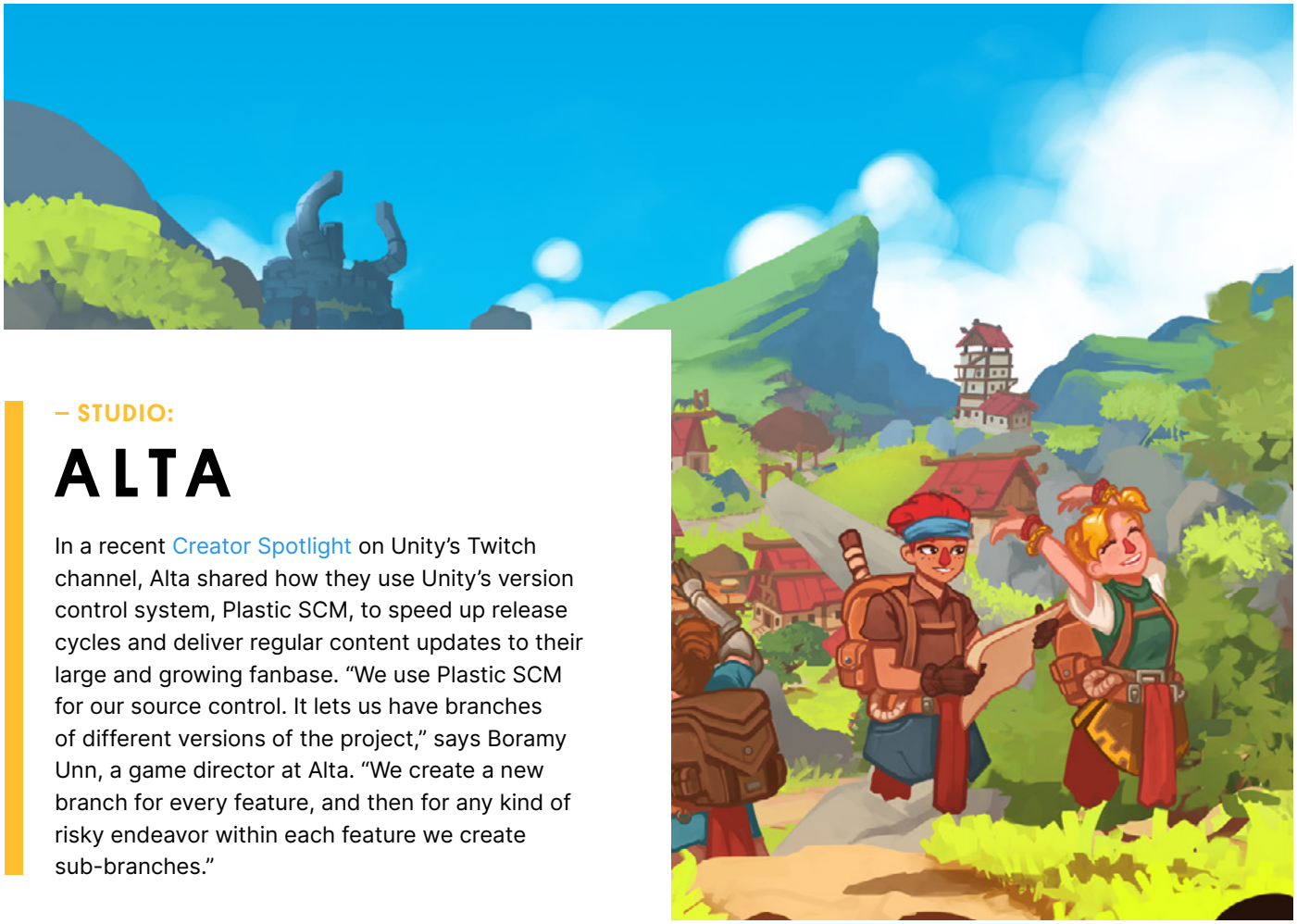
Of all the DevOps practices for game development environments, frequent, automated testing may be the most crucial. Players want new content regularly, and studios are under enormous pressure to meet launch dates with new, breakthrough features. But if one significant influencer has a bad experience on top of a bad day, the ramifications can be devastating. Getting QA right is imperative.



GAME STUDIOS MAKING IT WORK

DevOps and agile practices are largely just commonsense work habits. But when they're applied to games involving complex code, thousands of assets, and unforgiving performance constraints, studios need dedicated DevOps tools to ensure meticulous source control, continuous integration and delivery, safe and secure artifact management, and frequent, automated testing.

Using DevOps tools that integrate tightly with your development platform compounds their effectiveness. Below are just a few of the thousands of Unity creators using our growing library of DevOps tools to improve their workflows and produce stunning games.



— STUDIO:

ALTA

In a recent Creator Spotlight on Unity's Twitch channel, Alta shared how they use Unity's version control system, Plastic SCM, to speed up release cycles and deliver regular content updates to their large and growing fanbase. "We use Plastic SCM for our source control. It lets us have branches of different versions of the project," says Boramy Unn, a game director at Alta. "We create a new branch for every feature, and then for any kind of risky endeavor within each feature we create sub-branches."

A Township Tale by Alta

— STUDIO:

KO_OP

KO_OP is a cooperatively owned game studio based in Montreal. They were searching for a version control solution that would remove roadblocks between artists and engineers as they worked on their latest project, *Goodbye Volcano High*. "Plastic Gluon was a delightful surprise," said Jacob Blommestein, a developer at KO_OP, in a recent Unity case study. "The artists didn't have to know anything about branching or merging. They just added their .psd files, and the versioning was transparent."

Goodbye Volcano High by KO_OP

Return to Nangrim by Sycoforge

— STUDIO:

SYCOFORGE

In a recent interview on the Unity blog, CEO Michela Rimensberger shared how Plastic SCM is helping Sycoforge and their community work together to bring their ambitious game *Return to Nangrim* to life. “From a technical perspective, Plastic SCM has been a lifesaver for collaboration,” she said. “We tried out many different versioning tools and quickly realized that none of them really suited the needs of a gamedev company. While some were usable for programmers, artists had sweat outbreaks using them. With Plastic, we found a real game changer. It’s easy to use for both non-programmers and programmers.”

Quantum League by Nimble Giant Entertainment

— STUDIO:

CRYPTIC STUDIOS

Perfect World subsidiary Cryptic Studios uses Backtrace to automate bug and error tracking. “Backtrace has changed the way we explore and use crash data,” said Cryptic Studios senior producer Brent Lamb in a Unity e-book released this year. “Everyone feels comfortable running queries and understanding the health of one of our games.”

GET STARTED TODAY

Speed up your workflows and build out your DevOps toolchain with Unity's DevOps solutions. Create more efficiently to get to market on time, at the quality you expect. Explore solutions here.


Unity®