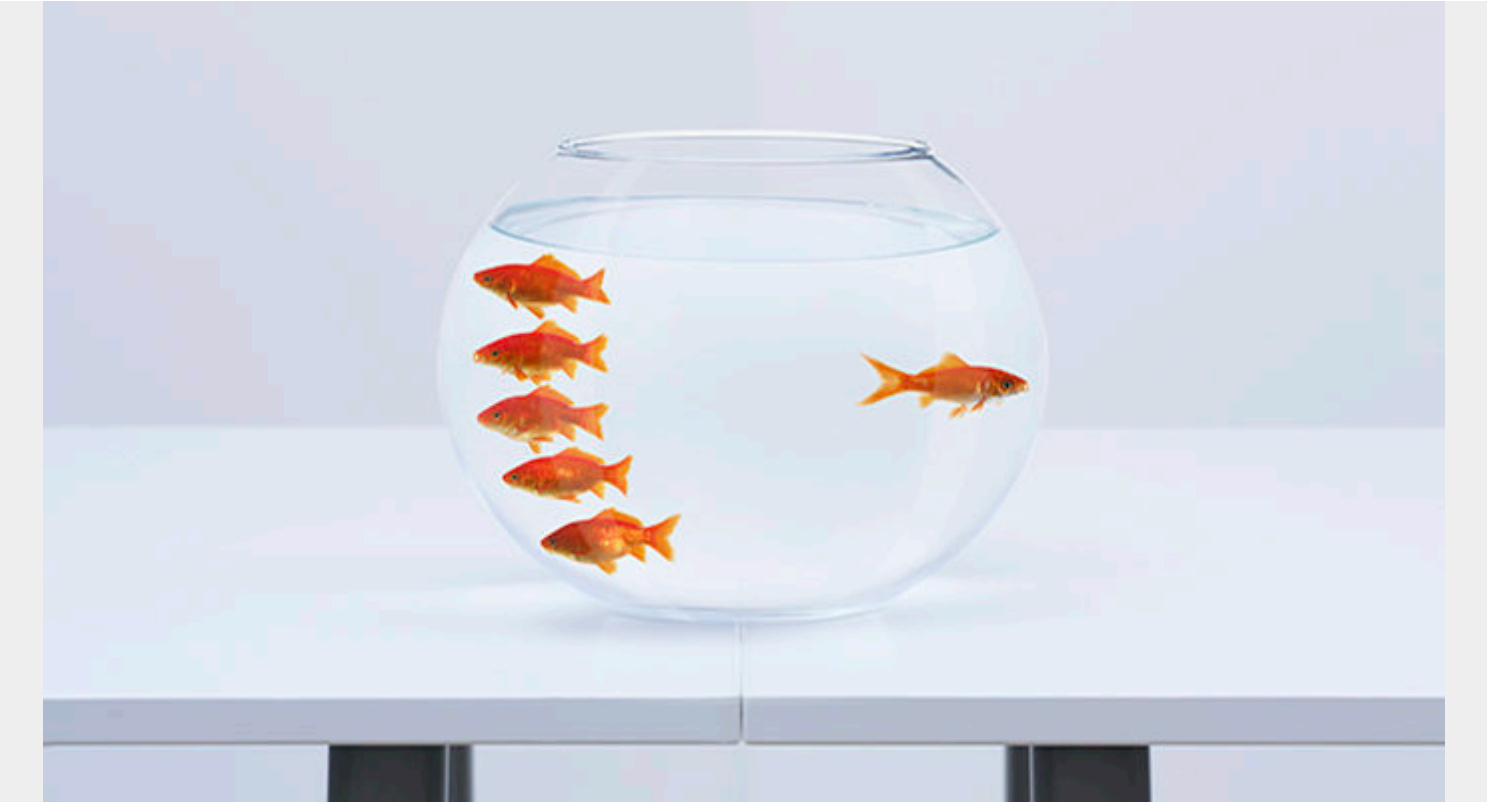
PATCH VS HOTFIX VS COLDFIX VS BUGFIX: DIFFERENCES EXPLAINED



Programmers have to be on their A-game to fix bugs, especially once the software is live and people are actively using it. Depending on the type of bug, you'll have to decide the best way to debug it—with the least amount of impact to the user experience.

Things get even more urgent when a security vulnerability is discovered, and it's all-hands-on-deck until a solution is implemented that will prevent successful exploitation of the system's weakness, removing or mitigating a threat's capability.

Finding and fixing the problem is just the first step. Then, you have to decide how to fix the problem—and how to

roll it out in a way that minimizes the impact to users. That remedy might be delivered via:

- Patch
- Hotfix
- Coldfix
- Bugfix

Those terms are often used interchangeably, but there are differences in each one based on how a programmer incorporates their solution into the software.

What's a patch?

In the early days of computing, a patch was, quite literally, a patch. Analog computers used punched cards and paper tapes to input programs the machines used for performing their calculations. These "decks" contained rows of holes and spaces that were a computer's software, and just like today, the software suppliers would need to make changes to the programming.

These updates were distributed on smaller pieces of paper tape or punched cards, and the recipients were expected to cut the bad part of the code out of the deck and patch in the replacement segment—hence the name.

Of course, patching has come a long, digital way. Patches for today's computers typically update existing software version's code by modifying or replacing it using a publicly released executable program.

Patches are often temporary fixes between full releases of software packages. Patches are used to correct both large and small issues that may or may not require immediate attention, such as:

- Fixing a software bug
- Installing new drivers
- Addressing new security vulnerabilities
- Addressing software stability issues
- Upgrading the software

Generally, patches are automatic updates that self-install packages in various sizes, from a few kilobytes up to large patches, like those for Windows that can be over 100 Mb. And like any Windows user can confirm, the installation of certain patches (on Patch Tuesday, of course) can cause interruptions and downtimes while being installed and possibly even require a system restart or two.

Most patches are delivered on a set schedule. They can be included in the product's new version release with additional updates and fixes.

What's a hotfix?

Hotfixes can also solve many of the same issues as a patch, but it is applied to a "hot" system—a live system—to fix an issue:

- Immediately
- Without creating system downtimes or outages.

Hotfixes are also known as QFE updates, short for quick-fix engineering updates, a name that illustrates the urgency.

Normally, you'll create a hotfix quickly, as an urgent measure against issues that need to be fixed immediately and outside of your normal development flow. Unlike patches, hotfixes address very specific issues like:

- Adding a new feature, bug, or security fix
- Changing database schema

Importantly, hotfixes are not always publicly released, in contrast to patches.

Here's an example: Let's say a bank learns that their banking app could be hacked, exploiting and revealing user data like passwords, usernames, and account information. Even if the hack hadn't occurred yet, it's a risk so significant that even identifying its potential requires urgent action. The security team will likely drop everything, scrambling to deliver a hotfix that solves the vulnerability as soon as possible, with minimal disruption.

What's a coldfix?

Where a hotfix is executed quickly without restarting any systems or hardware, a coldfix is just the opposite. Implementing a coldfix requires users to log out of the software while entire systems need to be rebooted for fixes to go into effect.

These types of updates are common in online multiplayer games, for example, so they're normally announced several days ahead of time to give users advanced notice the service will be unavailable while the fix is completed. Notices generally include estimated times the service will be back online since outages can last from a few minutes to several hours depending on the update.

What's a bugfix?

We're all familiar with the term bug: a programming defect or glitch that creates errors within a system or software. Removing these bugs is a practice called debugging.

Even though the cute name makes these errors sound small and only mildly irritating, like a gnat, developers and programmers can spend a lot of time searching several different types of common errors, such as:

- Syntax or type errors
- Typos and other simple errors
- Implementation errors
- Logical errors

Implementing a bugfix, also known as a program temporary fix (PTF), can be as simple as adding missing parentheses in a piece of code. But the fix can become quite challenging if the symptoms don't clearly point to a cause.

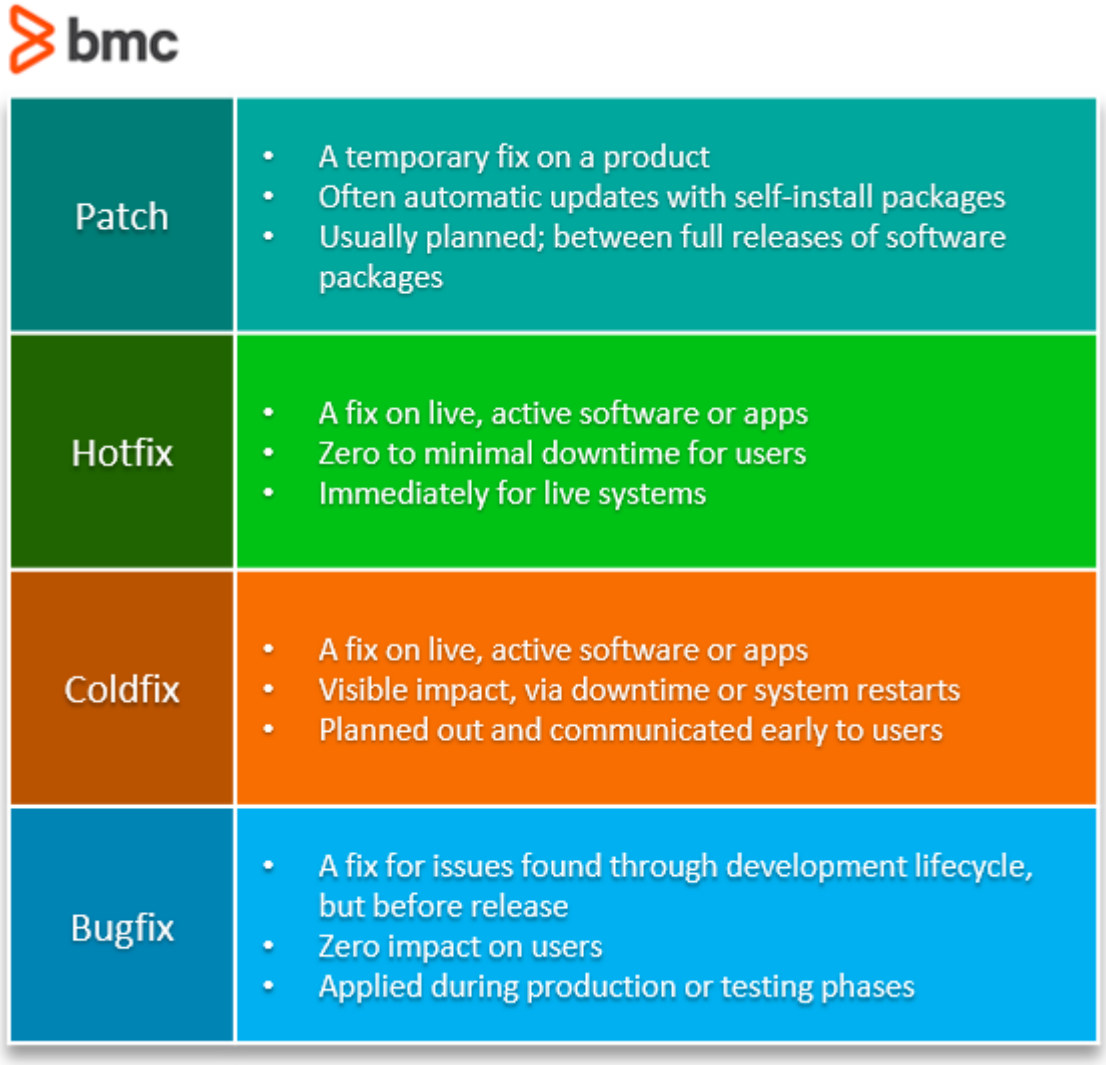
For instance, the cause and the symptom may be remote, with either located in the program code and the other in the execution of the program, or both.

Symptoms can also be difficult to reproduce for better understanding the problem. Once you've uncovered the root cause and issued a bug fix, however, it's not uncommon for your programmers

to find that one bugfix can actually introduce a new bug.

A bugfix sounds a lot like a hotfix, but the difference lies in the timing and execution of the correction. Bugfixes generally describe issues that are found and resolved during production or testing phases or even after deployment as part of the normal release cycle of a product. Hotfixes are applied only after the product has been released and is live.

Summing up the fixes



What are bug bounties?

As software increases in complexity, debugging before and after a product launches is vital for protecting your brand.

Applications are increasingly complex, multi-threaded, and large, with a greater number of developers working on them. All this complexity makes tracking down bugs more difficult and unpredictable. Multithreaded programs:

- Lengthen the time elapsed between the root cause of the bug and its detection.
- Make bugs difficult to track down and reproduce.

Bugs are a risk too big for you to ignore. Programmers will spend weeks hunting them or even offer bug bounties to get help finding the problems in their code before they can apply the right fix.

How to avoid bugs

The only ways to avoid bugs and the time spent on fixing them are to write better code. And until everyone starts writing perfect code, we can expect a few more bugs to get into places they shouldn't be.

Related reading

- BMC DevOps Blog
- Patch Management: A Brief Introduction
- Error Budgets Explained: Risk & Reliability in One Metric
- Deployment Pipelines (CI/CD) in Software Engineering
- Deploying vs Releasing Software: What's The Difference?
- What Is Spaghetti Code?