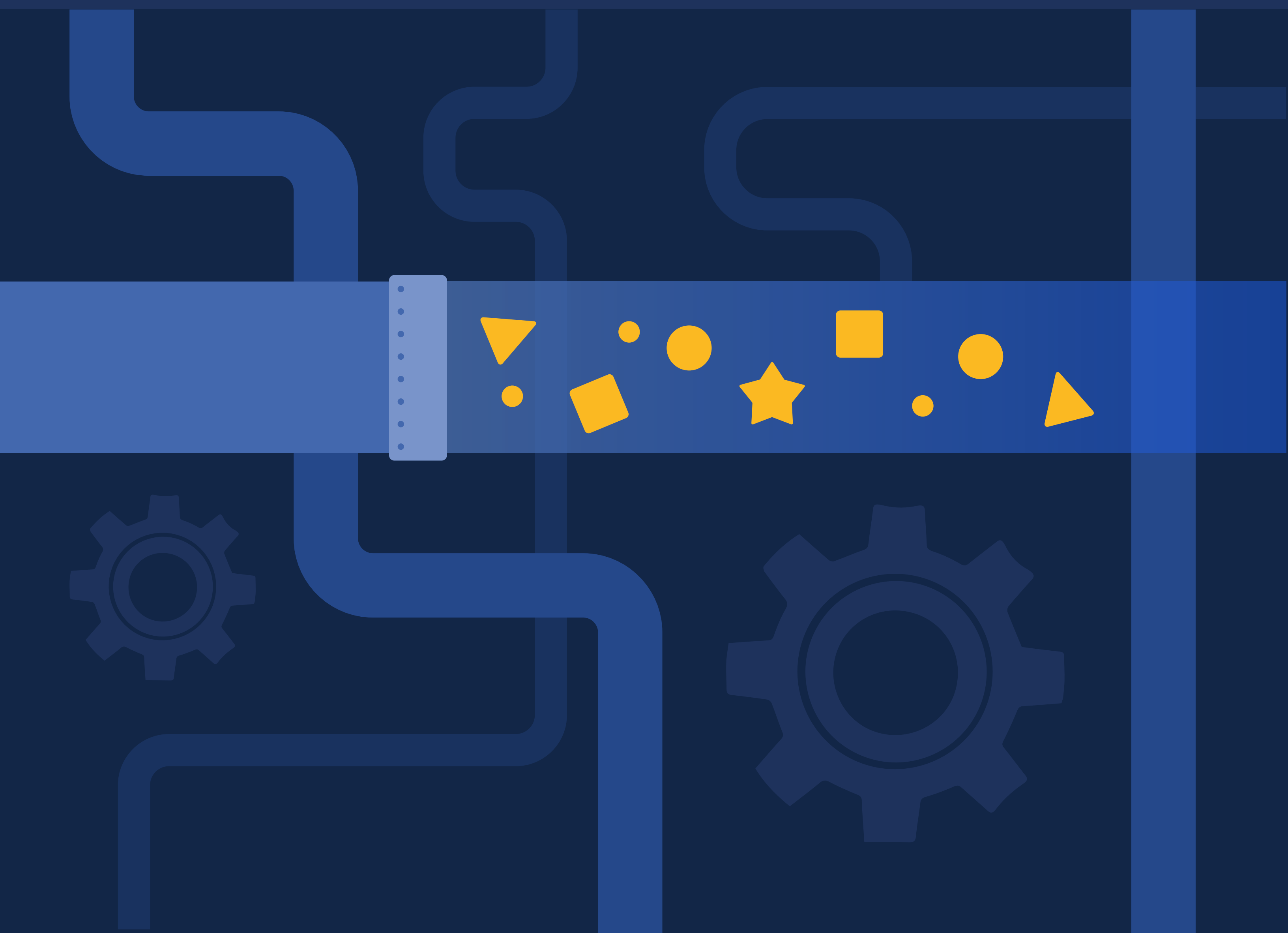
Mattermost

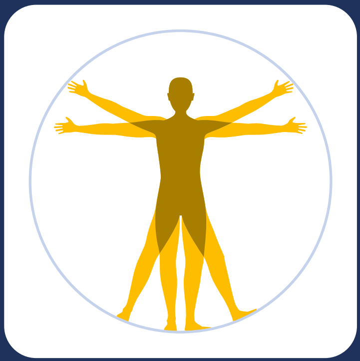
Level Up Your Release Management Workflow



What does your release management workflow look like?

Perhaps the most important process for any software team is their release management workflow. Taking releases from start to finish requires coordination across a broad range of teams and systems, and it's all too easy for something to slip through the cracks, slow down, or break.

In this guide, we'll cover what software development leaders need to know about developing and improving their own release management best practices so their teams can ship software releases faster and more predictably.



PART 1

Anatomy of a release management process

Before we dive into what a great release management process looks like, let's take a step back and consider the goals of release management. For most organizations, release management ensures that your team stays aligned across every stage of the software release lifecycle, from planning, to release, to monitoring, and back again. A well-executed release management process will:

- Manage any risks associated with the software release — from security concerns and bugs to usability issues and beyond
- Keep the team aligned on the status of the release
- Deliver software consistently and predictably
- Ensure customers and users have an optimal experience

Every team is unique, so customizing your release management plan to align with your team's tools and workflows is critical. No matter what your final product is, efficient and secure methods for releasing it do exist — you just have to find them. How? By honing your process in a collaborative environment with your team — and iterating on both victories and mistakes. Every time you release your product, your process will improve.

Keep in mind that your release management process is about more than the success of any one release. Over the long term, release management should help you achieve your DevOps goals. Implementing continuous integration of the product makes your software more robust and responsive to end users.

Stage 1: Planning

Release planning should be a cross-functional process that starts as early as possible. Keeping the team aligned at this stage is critical because it helps ensure everyone is on the same page throughout the entire software development *and* release management process. Giving everyone a single source of truth that they can use and reference from day one keeps conversations centralized and sets the groundwork for easier collaboration.

Release Planning Checklist

- [ ] Triage and check for pending tickets and PRs to merge
- [ ] Start drafting changelog, feature documentation, and marketing materials
- [ ] Review and update project dependencies as needed
- [ ] Have QA prepare release testing assignments
- [ ] Merge database upgrade

Stage 2: Release testing

There's nothing worse than releasing an update only to discover that you've introduced a bug, broken a critical workflow for your customers, or otherwise created more problems than you've solved. That's why a strong release testing practice is important to any release process. It's easy to feel like you can skip this step – or blaze through it quickly – but giving your QA team the time they need to appropriately test the build ensures the quality of the final product.

To keep release testing fast and effective, many teams stick to a set of smoke tests as part of their release process. Smoke tests confirm the validity and stability of the build, so it's important to find your team's testing sweet spot between "not enough to get a clear picture" and "so many that you slow down your release cadence." Many experts recommend running about 20–50 smoke tests that focus on core user flows and functionality.

Another essential part of this stage is knowing what to do with the results of your smoke tests. Having a clear process for triaging test results and resolving issues before release — as well as knowing what kinds of issues can wait until a future bug fix — will help your team clear quality hurdles quickly.

Release Testing Checklist

- ☐ Cut a release candidate
- ☐ QA runs smoke tests on the pre-release build
- ☐ QA runs automated load tests and upgrade tests on the pre-release build
- ☐ Triage and merge regression bug fixes

Stage 3: Prepare release for production

QA has approved the release — it's almost showtime! Keep the team aligned as all the boxes are checked and everything is confirmed; missing a step here can mean blind spots as you push the release. A shared playbook keeps the team on the same page and ensures their focus is in one place.

At this point, it's also time to start looping in internal teams and stakeholders. Including a checklist of need-to-know information for internal communication with all the essential details about the release will help them cascade out information to customers and team members, build supporting collateral, and do anything else they need to do to support the release. Having everything the larger team needs to publicize the release centralized and easy to find ahead of time will ensure a smoother launch for everyone.

Preparation Checklist

- ☐ QA final approves the release
- ☐ Cut the final release build and publish
- ☐ Notify internal teams and stakeholders

Stage 4: Publicize release

You've gotten over the biggest hurdle, and the release is out in the world and in the hands of real users. But no matter how useful, groundbreaking, or essential your release is, it won't make an impact if no one uses it. Communicating and publicizing your release is just as important as the technical work involved with launching it.

Many tasks at this stage benefit from having some amount of standardization. Providing templates makes life easier for your team as they spread the word about your release, from guidelines for feature documentation to customizable email templates.

Publication Checklist

- ☐ Deploy changelog, upgrade notes, and feature documentation
- ☐ Update release download links in relevant docs and webpages
- ☐ Publish announcements and marketing

Stage 5: Release follow-up & retrospective

Once the release is in production, your team will shift into operations and maintenance mode. Now, it's time to slow down, take a look back at what happened, and iterate your processes for the next time.

Don't forget to assign the specific people responsible for organizing the retrospective and enacting changes to the process! It's easy to overlook these tasks when everyone feels relieved that the release is live. Assigning clear owners will ensure that every last task gets wrapped up nicely.

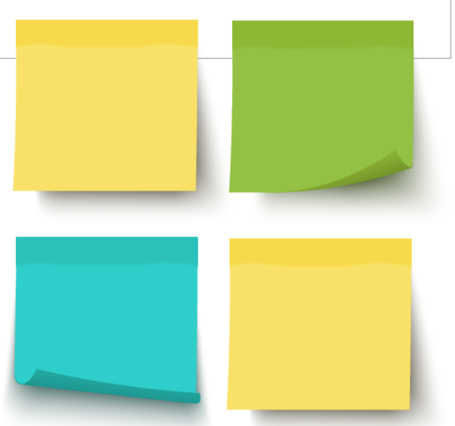
Check in with everyone involved in the release to see what went well and what can be improved next time. You might assume that your processes are fine because nothing went catastrophically wrong with a given release. In reality, incremental improvements can make for big gains over time. Retrospectives are a great opportunity to check in with your team, tools, and processes and make tweaks that will pay off in the long run.

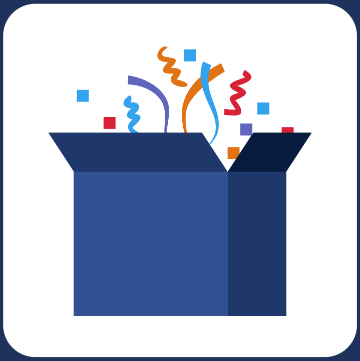
In order to have a successful retrospective, post-release metrics need to be gathered and discussed. Release management tools that monitor release metrics — time spent on the release, for example — will help you get a better understanding of your team's performance over time and are especially useful at this stage.

At last, you've reached the final step: updating your playbook for next time. Lessons learned don't help if they aren't documented and integrated into your processes moving forward.

Post-Release & Retrospective Checklist

- [ ] Schedule a release retrospective
- [ ] Compose release metrics
- [ ] Archive the release channel and create a new one for the next release
- [ ] Hold a retrospective
- [ ] Update your release management playbook





PART 2

Best tools for release management

You likely already have many of the tools you'll need for release management as part of your organization's tool stack. The key is understanding the role they play in your processes — and knowing how to bring them together effectively.

Communication and collaboration tools

Why It's Important: Internal and external comms can quickly get confusing; having clear channels for internal communication keeps the team aligned and prevents side conversations that can silo key information away from the main group. Collaboration tools can also be powerful platforms for centralizing other tools and processes into a single source of truth.

Don't forget about external communication, either! If your organization uses marketing automation or email software to communicate with customers, knowing who within the organization can send emails — and what the process is for doing so — will streamline communications when time is of the essence.

CI/CD tools

Why It's Important: DevOps processes are built on the backs of tools designed for CI/CD, such as continuous integration servers. They contain a wealth of information about release candidate status, product health that your team relies on to ensure a successful release. These tools are central to automating, streamlining, and speeding up release processes, but switching between them can create friction in your release workflow.

Get more out of your release management tools with ChatOps

Setting up a ChatOps practice that lets your team kick off builds, check on the status of smoke tests, and more reduces context switching during your release process and keeps your team's timeline on time.

Want to learn more about getting started and scaling ChatOps? Check out our Guide to ChatOps for Enterprise Teams.

QA testing tools

Why It's Important: QA teams play an essential role in the release management process — so it's important for them to have testing tools to ensure that they can get the job done efficiently! Test automation is an important component of release testing as it allows testing to be done more rapidly.

Deep integration of testing tools with your collaboration platform allows your QA team to run and review test results more effectively while keeping the rest of the team looped into the process.

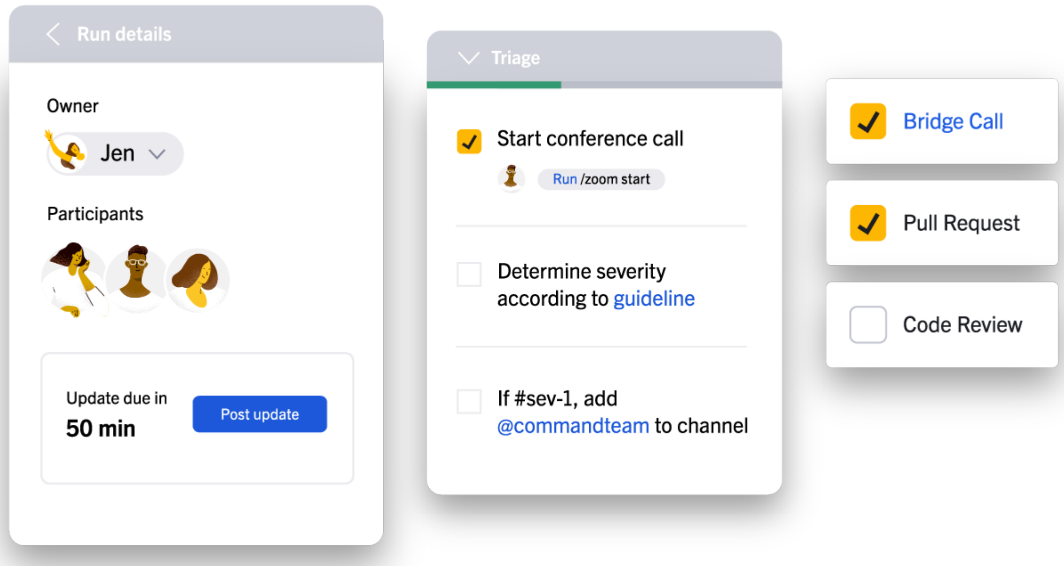
Observability and data analytics tools

Why It's Important: Software development often happens across different tools and systems, but everything needs to come together for a release to go off without a hitch. Integrating different sources of data ensures your team has all the information they need at their fingertips without having to dig through a dozen different tools.

Process documentation or runbook tools

Why It's Important: The best release management process is worthless if your team can't reference it quickly and effectively. A playbook that codifies processes and keeps the next steps in front of the team is essential to any release management stack.

Playbooks that are integrated into the tools your team already uses for normal daily operations and communications are ideal because they keep release-related conversations centralized and discoverable for the team throughout the entire release cycle.



PART 3

What teams get wrong about release management workflows

Want to improve your release management workflows? Avoid these all-too-common release management mistakes that can lead to slower releases, missed deadlines, and more.

Not budgeting enough time for the release process

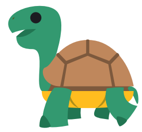
We've all heard the golden rule of not releasing on a Friday; at the heart of this rule is the need to give your team the time and resources needed to do the release right.

Accurately estimating project and task timelines is challenging, but those estimates are often used as the basis for the deadlines and release schedules that your team will be held to down the line. When your team doesn't have enough time to work through their release management workflow, deadlines can slip — or worse, quality can suffer.

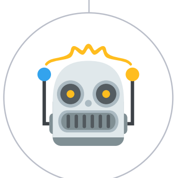
Poor release definition

Budgeting enough time for your entire release process starts with knowing exactly what will be released and when. Good communication and collaboration practices help keep the scope of the release clear for every stakeholder — so when it comes time to launch, no one is left wondering what happened to that one *particular* update they thought was going to be available with this release. Incorporating regular updates into your release planning and preparation process keeps everyone on the same page — especially when something changes.

Overly manual processes



Every team has processes that must be done manually. But manual tasks take time and can throttle your team's ability to release software quickly. The more frequently your team releases software, the more time they'll waste on manual tasks — leaving them with less time for everything else. Release management workflows are a great opportunity to use automation since each release will have consistent tasks that must be completed.



Start automating your CI/CD workflows

Ready to start taking those manual tasks off your release to-do list? Learn how to integrate your GitLab CI/CD pipelines into Mattermost in this tutorial.

Lack of visibility and clear communication

The fastest way to slow down a release? Keep your team in the dark about what's going on. Information silos are one of the biggest challenges that developers face when it comes to collaborative projects, and release management workflows have more than their fair share of complex collaboration needs.

PART 4

How to improve release management processes

So now that you know what *not* to do as you perfect your release management plan, what should your team do to continue to optimize your release management practices? Here are some of the release management best practices that will help take your software releases to the next level.

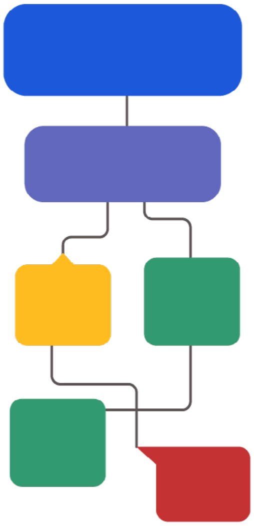
Unify tooling and increase visibility

Your release management process spans multiple tools and teams, so the more you can do to keep essential information discoverable, the faster and more effectively your team will be able to manage your release process. Bringing all of the tools in your release workflow together prevents information silos and gives your team a fuller picture of what's going on during a release, which can help you spot any potential problems early on.

Improve process documentation

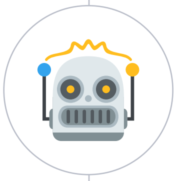
Process documentation can vary from organization to organization and process to process. Improving your process documentation to have enough detail to give your team a clear roadmap for every release — while still giving them enough room to adapt and evolve the process to the specific needs of a given release — ensures that releases stay predictable and stress-free.

Automate repeated workflows



Every release will have repetitive tasks. Taking those tasks off your team's plate and handing them over to automated systems can give your team back valuable time to ensure the release goes smoothly.

Take stock of your release management workflows and look for areas where your team spends time on the same task over and over. Those tasks might be great candidates for automation. Remember, even small tasks can add up to a large time sink in the long run and can be a great starting point for workflow automation!



Get some workflow automation inspiration

To keep their weekly releases streamlined, the team at n8n uses a number of custom slash commands, bots, and tool integrations in their Mattermost environment. Check out their story for automation inspiration, from bots that allow developers to push releases to production directly from their chat window to slash commands that help technical writers stay on top of documentation updates.

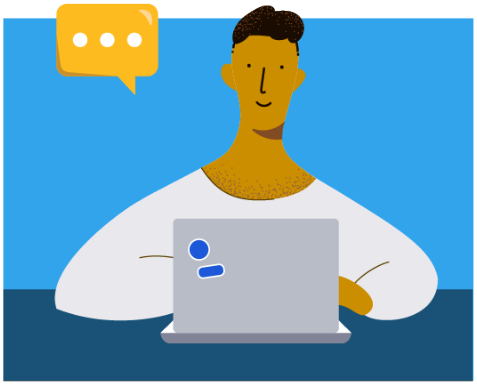
Start the process earlier

Think about your last release. Did any critical tasks feel rushed? Were there times when members of the team weren't sure what needed to happen next? Many of the challenges of release management can be addressed by giving your team more breathing room to do their work, but that doesn't mean that you need to slow down your release cadence! Instead, shift your perspective about when a release begins and start the earliest stages as soon as possible.

Always do your retrospective

Retrospectives can be easy to dismissas a nice-to-have step in the release management process. By the time your team has checked all the boxes of a release, navigated any un-foreseen challenges, and celebrated their wins, they're ready to move on to the next project.

But the little issues that put a single release off-kilter can turn into major problems over time, and retrospectives help your team course-correct and make improvements that will keep your processes moving forward. Treat your retrospective as an essential stage to reap the full benefits of self-reflection!



PART 5

Final thoughts on release management

Getting releases right can be challenging, but it's essential for any software development team. We hope this guide has helped you reflect on and refine your release management practices and that you now understand how to improve your processes for faster, smoother release management in the future.

Essential takeaways for improving release management

Rethink Your Release Definition: Start planning for your next release as early as possible to give your team as much time as possible to do their jobs well. Assigning clear owners to each and every action item ensures that nothing slips through the cracks and everyone knows who's responsible for what.

Keep Everything Together: Integrating systems, centralizing key documents and communication, and keeping process documentation where your team can reference it quickly helps keep the team aligned as they collaborate on every release.

Learn from Every Release: Whether a given release is smooth sailing or had a few hiccups along the way, there's always something your team can do better the next time around. Use every release as an opportunity to improve your processes for the next one.

Build your release management playbook in Mattermost

Ready to build a release management playbook for yourself — or want to see how using Mattermost Playbooks can help you supercharge your existing release management playbook? Create a free Mattermost environment then check out this tutorial on using Mattermost for release management to learn more.

Here's to creating an efficient and resilient release management playbook that enables your team to ship great releases every time!

About Mattermost

Hundreds of thousands of developers around the globe trust Mattermost to increase their productivity by bringing together team communication, task and project management, and workflow orchestration into a unified platform for agile software development.

Founded in 2016, Mattermost's open source platform powers over 800,000 workspaces worldwide with the support of over 4,000 contributors from across the developer community. The company serves over 800 customers, including Samsung, Nasdaq, SAP, European Parliament, and the United States Air Force, and is backed by world-class investors, including Redpoint, YC Continuity, Battery Ventures, and S28 Capital. To learn more, visit www.mattermost.com.

Mattermost

Mattermost.com

🎉 9