

Feature Lifecycle

For teams that follow a structured build and release cycle, having a reliable, shared workflow makes the difference between chaos and consistency. With every new feature in development the team needs to know what the specs are, how it fits in the roadmap, what the customer feedback was, where to find the repository, who is responsible for each step, and so much more. That information can be scattered and buried in documentation and handbooks, requiring digging and institutional memory just to get the basics for every sprint.

Reusable runbooks and standardized templates bring order to the chaos. The Mattermost platform gives teams a shared space to communicate, collaborate, and concentrate on all stages of the software development lifecycle.

In this guide, we'll show you how to customize a pre-built Mattermost [Playbooks](#) template to create your own reusable process playbook. You'll then be able to create "runs" from your custom playbook for each iteration of your process in the future. During each run, you'll collect insights, metrics, and retrospectives to reference, building centralized institutional knowledge to strengthen the entire team.

What you'll need

- ✓ An understanding of your current feature development process. You may have [an established and well-documented process](#), no process at all, or something in between. Wherever you are, we'll show you some options and structures you can build on.
- ✓ A Mattermost server. You can test this out on our [open source community server](#), your own server, or start a [free cloud trial](#).
- ✓ The "Feature Lifecycle" Playbooks template from Mattermost, which can be found inside Mattermost at the bottom of this screen <https://{hostname}.mattermost.com/playbooks/playbooks> or [in the marketplace](#) as an [importable json file](#).

About the Feature Lifecycle template

The screenshot shows the Mattermost web interface with the 'Playbooks' tab selected. The title 'Feature Lifecycle Template' is displayed with a dropdown arrow and a lock icon. Below the title, a sub-header says 'Customize this playbook to reflect your own feature lifecycle process.' There are three tabs at the top: 'Usage', 'Outline' (which is selected), and 'Reports'. On the left, there's a sidebar with a 'CONTENTS' section containing links to 'Summary', 'Status Updates', 'Checklists', 'Retrospective', and 'Actions'. The main content area is divided into sections: 'Summary', 'Status Updates', and 'Checklists'. The 'Summary' section contains a 'One-liner' (described as enabling users to prescribe a description template), 'Targets release' (with bullet points for code complete and customer release dates), and 'Resources' (links to Jira Epic, UX prototype, Technical design, and User docs). The 'Status Updates' section specifies a status update every 1 day to no channels and no outgoing webhooks.

All Playbooks in Mattermost have the same set of features, but how you fill out the details varies depending on the process you are building a playbook for. For the Feature Lifecycle template, you'll find some process specific suggestions out of the box for the run summary and checklists.

Your team will especially want to modify automations and checklists in order to use this template to create a custom playbook from which you will create regular runs:

Template: a simple and generic starting point

↳ **Playbook:** the source copy of your repeatable process

↳ **Run:** a single instance of your repeatable process

The Feature Lifecycle template is designed to help a team follow the same set of steps for every new feature or feature update in development. The checklists include these sections:

Plan – Work together in channels or calls to define the problem space, solutions, and set a timeline for the work.

Kickoff – Identify owners and contributors and set a schedule for check-in meetings.

Build – Break down the actual work and add more items to the checklist, align on effort and capacity.

Ship – Work with other teams like documentation and marketing to plan for the release and measurement of your feature.

Follow up – Don't ship and forget! Set your team up for success with review of reporting, discussion of what went well and what should change, and plan to update your playbook to reflect your learnings.

For example, the Mattermost team uses a variation on this template for our own playbook, [“Playbooks Team: Feature Swimlane,”](#) which you may view live on our community server.

How to use and modify the Feature Lifecycle template

In Mattermost, navigate to Playbooks using the global menu in the upper left corner. Select the Playbooks button next to Runs. Scroll down if needed to bring a section titled “Do more with Playbooks” into view. Select the “Feature Lifecycle” template.

OR, to download and install any PB template from github.com/mattermost/mattermost-product-templates (or a friend) you may [copy, export and import playbooks as .json files.](#)

Once you are looking at the template in Mattermost, you'll be able to start modifying it to match your own process. Start with the title! Here are some other areas you might consider making changes:

Update the playbook description so teams know when to use the playbook and who should use the playbook.

Update the run summary template to include placeholders for reference links that make sense for your team. This can include a template statement describing the feature (ie “as a __ I want __ so that __” or similar), target dates for release to various environments, links to analytics, GitHub/GitLab repositories, designs and more.

Set a cadence, destinations, and template for regular status updates. You may want to send updates to other channels in Mattermost, or to outgoing webhooks. These can be [changed during a run](#), in case the situation evolves.

Update the checklists. Add sections, rename things, add checklist items, and for a real power move, add slash commands. Everything accepts markdown so feel free to include detail and links.

Define your key metrics in the retrospective section – How will you measure the success of your process? Our team measures “Code Complete Estimation Accuracy” as an integer type metric. The other options are dollars and duration.

Set up a retrospective template. This is a text field your team will fill in at the end of a run, following a template so that it’s easy to browse and skim retros in bulk later on. Specific cues like “How many big changes did we introduce after initial planning and why” can help identify areas for improvement on future runs.

Set [channel actions](#) to automatically invite the same people or teams for every run of your playbook, send an outgoing webhook when a run starts, and keep your sidebar tidy by categorizing the run channel.

Learn more about using Playbooks

Playbooks can be an incredibly powerful tool for your team over time. As you develop your custom playbooks you’ll gain more and more benefit both from having all the info the team needs in one place and having the metrics and analytics to share your success.

Read a whole lot more in the [Mattermost user guide to Playbooks »](#)

