Managing a merge queue

In this article

About merge queues
Configuring continuous integration (CI) workflows for merge queues
Managing a merge queue
How merge queues work
Further reading

You can increase development velocity with a merge queue for pull requests in your repository.

Who can use this feature
People with admin permissions can manage merge queues for pull requests targeting selected branches of a repository.

Pull request merge queues are available in any public repository owned by an organization, or in private repositories owned by organizations using GitHub Enterprise Cloud. For more information, see "GitHub’s plans."

About merge queues

A merge queue helps increase velocity by automating pull request merges into a busy branch and ensuring the branch is never broken by incompatible changes.

The merge queue provides the same benefits as the **Require branches to be up to date before merging** branch protection, but does not require a pull request author to update their pull request branch and wait for status checks to finish before trying to merge.

Using a merge queue is particularly useful on branches that have a relatively high number of pull requests merging each day from many different users.

Once a pull request has passed all required branch protection checks, a user with write access to the repository can add the pull request to the queue. The merge queue will ensure the pull request's changes pass all required status checks when applied to the latest version of the target branch and any pull requests already in the queue.

A merge queue may use GitHub Actions or your own CI provider to run required checks on pull requests in a merge queue. For more information, see "GitHub Actions documentation."

For more information on merging a pull request using a merge queue, see "Merging a pull request with a merge queue."

Configuring continuous integration (CI) workflows for merge queues

Notes:

- A merge queue cannot be enabled with branch protection rules that use wildcard characters (`*`) in the branch name pattern.
- A merge queue will wait for required checks to be reported before it can proceed with merging. You must update your CI configuration to trigger and report on merge group events when requiring a merge queue.

Triggering merge group checks with GitHub Actions

You **must** use the `merge_group` event to trigger your GitHub Actions workflow when a pull request is added to a merge queue.

Note: If your repository uses GitHub Actions to perform required checks on pull requests in your repository, you need to update the workflows to include the `merge_group` event as an additional trigger. Otherwise, status checks will not be triggered when you add a pull request to a merge queue. The merge will fail as the required status check will not be reported. The `merge_group` event is separate from the `pull_request` and `push` events.

A workflow that reports a check which is required by the target branch's protections would look like this:

```
on:
  pull_request:
  merge_group:
```

For more information on the `merge_group` event, see "Events that trigger workflows."

Triggering merge group checks with third-party CI providers

With third-party CI providers, you will need to update your CI configuration to run when a branch that begins with the special prefix `gh-readonly-queue/{base_branch}` is pushed to. These are the temporary branches that are created on your behalf by a merge queue and contain a different `sha` from the pull request.

Managing a merge queue

Repository administrators can require a merge queue by enabling the branch protection setting "Require merge queue" in the protection rules for the base branch. For more information, see "Managing a branch protection rule."

Once you have enabled the "Require merge queue" setting, you can also access the following settings:

- **Merge method**: Select which method to use when merging queued pull requests: merge, rebase, or squash.
- **Build concurrency**: The maximum number of `merge_group` webhooks to dispatch (between `1` and `100`), throttling the total amount of concurrent CI builds. This affects the velocity of merges that a merge queue can complete.
- **Only merge non-failing pull requests**: This setting determines how a merge queue forms groups of pull requests to be merged.

Enabled?	Description
Yes	All pull requests must satisfy required checks to be merged.

No	Pull requests that have failed required checks can be added to a group as long as the last pull request in the group has passed required checks. If the last pull request in the group has passed required checks, this means that the checks have passed for the combined set of changes in the merge group. Leaving this checkbox unselected can be useful if you have intermittent test failures, but don't want false negatives to hold up the queue.

- **Status check timeout**: Choose how long the queue should wait for a response from CI before assuming that checks have failed.

- **Merge limits**: Select the minimum and maximum number of pull requests to merge in a single group (between `1` and `100`), and a timeout after which the queue should stop waiting for more entries and merge with fewer than the minimum number of pull requests. Exactly how many PRs are included in a group depends on a merge queue's settings:

Merge Limit	Use Case
Maximum pull requests to merge	You can specify a maximum group size, which is useful if merges to your base branch trigger a deployment, and you want to make sure you’re not deploying too many changes at once.
Minimum pull requests to merge	You can specify a minimum group size, which is useful if merges to your base branch trigger a lengthy CI build or deploy process, and you don’t want to hold up the following entries in the queue.
Wait time	You can specify a timeout for reaching the minimum group size, which allows smaller groups to merge if there are no more PRs queued within your specified time limit.

How merge queues work

As pull requests are added to the merge queue, the merge queue ensures that they are merged in a first-in-first-out order where the required checks are always satisfied.

A merge queue creates temporary branches with a special prefix to validate pull request changes. When a pull request is added to the merge queue, the changes in the pull request are grouped into a `merge_group` with the latest version of the `base_branch` as well as changes from pull requests ahead of it in the queue. GitHub Enterprise Cloud will merge all these changes into the `base_branch` once the checks required by the branch protections of `base_branch` pass.

For information about merge methods, see "About pull request merges."

Successful CI

When multiple pull requests are added to the merge queue and when the temporary `merge_group` branches have successful CI results, they are both merged. In the following scenario, two pull requests are successfully added to the queue and merged to the target branch.

1. User adds pull request #1 to the merge queue.
2. The merge queue creates a temporary branch with the prefix of `main/pr-1` that contains code changes from the target branch and pull request #1. A `merge_group` webhook event of type `checks_requested` is dispatched and the merge queue will await a response from your CI provider.
3. User adds pull request #2 to the merge queue.
4. The merge queue creates a temporary branch with the prefix of `main/pr-2` that contains code changes from the target branch, pull request #1, and pull request #2, and dispatches webhooks.
5. When the GitHub Enterprise Cloud API receives successful CI responses for `merge_group` branches `main/pr-1` and `main/pr-2`, the temporary branch `main/pr-2` will be merged in to the target branch. The target branch now contains both changes from pull request #1 and #2.

Failing CI

After grouping a pull request with the latest version of the target branch and changes ahead of it in the queue, if there are failed required status checks or conflicts with the base branch, the pull request will be removed from the queue. The pull request timeline will display the reason why the pull request was removed from the queue.

The following scenario outlines what happens when a CI reports a failing status about one pull request.

1. User adds pull request #1 to the merge queue.
2. The merge queue creates a temporary branch with the prefix of `main/pr-1` that contains code changes from the target branch and pull request #1. A `merge_group` webhook event of type `checks_requested` is dispatched and the merge queue will await a response from your CI provider.
3. User adds pull request #2 to the merge queue.
4. The merge queue creates a temporary branch with the prefix of `main/pr-2` that contains code changes from the target branch, pull request #1, and pull request #2, and dispatches webhooks.
5. When the GitHub Enterprise Cloud API receives a failing status for `main/pr-1`, the merge queue automatically removes pull request #1 from the merge queue.
6. The merge queue recreates the temporary branch with the prefix of `main/pr-2` to only contain changes from the target branch and pull request #2.
7. When the GitHub Enterprise Cloud API receives successful CI responses for `merge_group` branch `main/pr-2`, the temporary branch `main/pr-2` will be merged in to the target branch without pull request #1 included.

There are a number of reasons a pull request can be removed from a merge queue:

- Configured CI service is reporting test failures for a merge group
- Timed out awaiting a successful CI result based off the configured timeout setting
- User requesting a removal via the API or merge queue interface
- Branch protection failure that could not automatically be resolved

Jumping to the top of the queue

When adding a pull request to a merge queue, there is an option to move your pull request to the top of the queue.

> **Note:** Be aware that jumping to the top of a merge queue will cause a full rebuild of all in-progress pull requests, as the reordering of the queue introduces a break in the commit graph. Heavily utilizing this feature can slow down the velocity of merges for your target branch.

The following scenario outlines what happens when a user jumps the queue.

1. User adds pull request #1 to the merge queue.
2. The merge queue creates a temporary branch with the prefix of `main/pr-1` that contains code changes from the target branch and pull request #1. A `merge_group` webhook event of type `checks_requested` is dispatched and the merge queue will await a response from your CI provider.
3. User adds pull request #2 to the merge queue.
4. The merge queue creates a temporary branch with the prefix of `main/pr-2` that contains code changes from the target branch, pull request #1, and pull request #2, and dispatches webhooks.
5. User adds pull request #3 to the merge queue with the jump option which introduces a break in the commit graph.
6. The merge queue creates a temporary branch with the prefix of `main/pr-3` that contains code changes from the target branch and pull request #3, and dispatches webhooks.
7. The merge queue recreates the temporary branches with the prefix of `main/pr-1` and `main/pr-2` that contain the changes from pull request #3, and dispatches webhooks.

Further reading

- "Merging a pull request with a merge queue"
- "About protected branches"