Filtering projects

In this article

Use filters to choose which items appear in your project's views.

You can customize which items appear in your views using filters for item metadata, such as assignees and the labels applied to issues, and by the fields in your project. You can combine filters and save them as views. For more information, see "[Managing your views](#)."

To filter a view, click $\equiv$ and start typing the fields and values you would like to filter for. As you type, possible values will appear. You can also open the project command palette, by pressing `Command` + `K` (Mac) or `Ctrl` + `K` (Windows/Linux), and type "Filter by" to choose from the available filters.
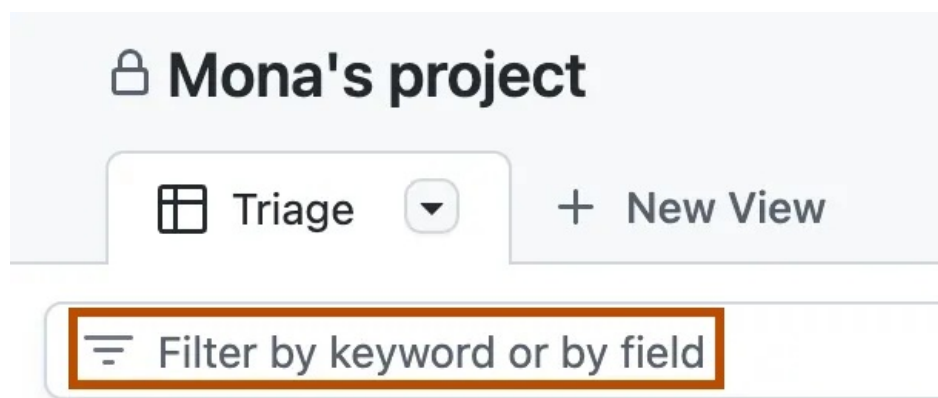
In board layout, you can click on item data to filter for items with that value. For example, click on an assignee to show only items for that assignee. To remove the filter, click the item data again.

Using multiple filters will act as a logical AND filter. For example, `label:bug status:"In progress"` will return items with the `bug` label and the "In progress" status. You can also provide multiple values for the same field to act as a logical OR filter. For example, `label:bug,support` will return items with either the `bug` or `support` labels. Projects does

not currently support logical OR filters across multiple fields.

The same filters are available for charts you create using insights for Projects, allowing you to filter the data used to create your charts. For more information, see "[About insights for Projects](#)."

When you filter a view and then add an item, the filtered metadata will be applied to new item. For example, if you're filtering by `status:"In progress"` and you add an item, the new item will have its status set to "In progress."

You can use filters to produce views for very specific purposes. For example, you could use `assignee:@me status:todo last-updated:5days` to create a view of all work assigned to the current user, with the "todo" status, that hasn't been updated in the last five days. You could create a triage view by using a negative filter, such as `no:label no:assignee repo:octocat/game`, which would show items without a label and without an assignee that are located in the `octocat/game` repository.

Filtering for fields 🔗

Qualifier	Example
`assignee:`*USERNAME*	**assignee:octocat** will show items assigned to @octocat.
`label:`*LABEL*	**label:bug** will show items with the "bug" label applied.
`field:`*VALUE*	**status:done** will show items with the "status" field set to "done."
`reviewers:`*USERNAME*	**reviewers:octocat** will show items that have been reviewed by @octocat.
`milestone:"`*MILESTONE*`"`	**milestone:"Beta release"** will show items assigned to the "Beta release" milestone.

Combining filters 🔗

You can create filters for multiple fields. Your view will show items that match all filters.

Qualifier	Example
`assignee:`*USERNAME* `field:`*VALUE*	**assignee:octocat priority:1** will show items assigned to @octocat that have a priority of **1**.

You can also filter for multiple values from the same field. If you separate the values with commas, your view will show items that match any of the provided values.

Qualifier	Example
`assignee:`*USERNAME*`,`*USERNAME*	**assignee:octocat,stevecat** will show items assigned to either @octocat or @stevecat.

To filter for multiple values from the same field but show items that match all of the provided values, you can repeat the qualifier for each value.

Qualifier	Example
`assignee:`*USERNAME* `assignee:`*USERNAME*	**assignee:octocat assignee:stevecat** will

show items that are assigned to both @octocat and @stevecat.

You can also combine filters that match some and match all items.

Qualifier	Example
`field:`*`VALUE`*`,`*`VALUE`* `assignee:`*`USER`* `assignee:`*`USER`*	**label:bug,onboarding assignee:octocat assignee:stevecat** will show items that have either the bug or onboarding labels but are assigned to both @octocat and @stevecat.

Negating a filter 🔗

You can invert any filter, including combinations, by prefixing with a hyphen.

Qualifier	Example
`-assignee:`*`USERNAME`*	**-assignee:octocat** will not show any items assigned to @octocat.
`-field:`*`VALUE`*	**-status:done** will not show any items with a status of "done."
`-field:`*`VALUE`*`,`*`VALUE`*	**-priority:1,2** will not show any items with a priority of either 1 or 2.

Filtering for items that are missing a value 🔗

You can use `no:` to filter for items that are missing a value

Qualifier	Example
`no:assignee`	**no:assignee** will show any unassigned items.
`no:reviewers`	**no:reviewers** will show pull requests that do not have a reviewer.
`no:`*`FIELD`*	**no:priority** will show items with an empty priority field.

You can also prefix a hyphen to negate this behavior and only return items that have a value.

Qualifier	Example
`-no:assignee`	**-no:assignee** will only show items that are assigned.
`-no:`*`FIELD`*	**-no:priority** will only show items that have a value in the priority field.

Filtering by item location 🔗

Use the `repo` qualifier to filter for items in a particular repository.

Qualifier	Example
`repo:OWNER/REPO`	**repo:octocat/game** will items in the "octocat/game" repository.

Filtering for item state or item type 🔗

You can use the `is` qualifier to filter for particular types of item or items in particular states.

Qualifier	Example
`is:STATE`	**is:open** will show open issues and pull requests.
	is:closed will show closed issues and pull requests.
	is:merged will show any merged pull requests.
`is:TYPE`	**is:issue** will show only issues.
	is:pr will show only pull requests.
	is:draft will show draft issues and draft pull requests.
	is:issue is:open will show open issues.

Filtering by close reason 🔗

You can filter closed items by their close reason.

Qualifier	Example
`reason:CLOSE REASON`	**reason:completed** will show items closed because they were completed.
	reason:"not planned" will show closed items with the "not planned" reason.
	reason:reopened will show items that have been reopened after previously being closed.

Filtering by the tracked-by field 🔗

You can filter for issues that are tracked by another issue in a tasklist. For more information, see "[Creating a tasklist](#)."

Qualifier	Example
`tracked-by:"OWNER/REPO#ISSUE NUMBER"`	**tracked-by:"octocat/game#11"** will show any issue tracked by a tasklist in issue #11 of the `octocat/game` repository.

Filtering for when an item was last updated 🔗

You can use the `{number}days` syntax to filter for when items were last updated.

Qualifier	Example
`last-updated:NUMBERdays`	**last-updated:1day** will show items last updated one or more days ago.
	last-updated:7days will show items last updated seven or more days ago.
	-last-updated:10days will show items that have been updated in the last ten days.

GitHub Enterprise Cloud marks an issue or pull request as updated when it is:

- Created
- Reopened
- Edited
- Commented
- Labeled
- Assignees are updated
- Milestones are updated
- Transferred to another repository

Filtering number, date, and iteration fields 🔗

You can use `>`, `>=`, `<`, and `<=` to compare number, date, and iteration fields. Dates should be provided in the `YYYY-MM-DD` format.

Qualifier	Example
`field:>VALUE`	**priority:>1** will show items with a priority greater than 1.
`field:>=VALUE`	**date:>=2022-06-01** will show items with a date of "2022-06-01" or later.
`field:<VALUE`	**iteration:<"Iteration 5"** will show items with an iteration before "Iteration 5."
`field:>=VALUE`	**points:<=10** will show items with 10 or less points.

You can also use `..` to filter for an inclusive range. When working with a range, `*` can be supplied as a wildcard operator.

Qualifier	Example
`field:VALUE..VALUE`	**priority:1..3** will show items with a priority of 1, 2, or 3.
	date:2022-01-01..2022-12-31 will show items from the year 2022.
	points:*..10 will show items with an points value of anything up to and including 10.
	iteration:"Iteration 1".."Iteration 4" will show items in "Iteration 1", "Iteration 2",

Filtering assignees and reviewers using keywords 🔗

You can use the `@me` keyword to represent yourself in a filter.

Qualifier	Example
`field:@me`	**assignee:@me** will show items assigned to the signed-in user.
	-reviewers:@me will show items that have not been reviewed by the signed-in user.

Filtering iteration and date fields using keywords 🔗

You can use the `@previous` , `@current` , and `@next` keywords to filter for iterations relative to the current iteration. You can also use `@today` to filter for the current day.

Qualifier	Example
`field:@keyword`	**iteration:@current** will show items assigned to the current iteration.
	iteration:@next will show items assigned to the next iteration.
`field:@today`	**date:@today** will show items with their date set to the current day.

You can also use `>` , `>=` , `<` , `<=` , `+` , `-` , and `..` ranges with keywords.

Qualifier	Example
`field:@keyword..@keyword+n`	**iteration:@current..@current+3** will show items assigned to the current iteration and the next three iterations.
	date:@today..@today+7 will show items with a date set to today or the next seven days.
`field:<@keyword`	**iteration:<@current** will show items assigned to any iteration before the current iteration.
`field:>=@keyword`	**date:>=@today** will show items with a date set to today or later.

Filtering by text fields 🔗

You can filter by specific text fields or use a general text filter across all text fields and titles. When filtering with text that contains spaces or special characters, enclose your text in `"` or `'` quotation marks.

Qualifier	Example
`field:"TEXT"`	**title:"API deprecation"** will show items with titles that exactly match "API deprecation."

`field:`*TEXT*	**note:complete** will show items with a note text field that exactly match "complete."
TEXT	**API** will show items with "API" in the title or any other text field.
`field:`*TEXT* TEXT	**label:bug rendering** will show items with the "bug" label and with "rendering" in the title or any other text field.

You can also use a `*` as a wildcard.

Qualifier	Example
`field:`**TEXT**	**label:*bug*** will show items with a label that contains the word "bug."
`field:`*TEXT**	**title:API*** will show items with a title that begins with "API."
`field:`**TEXT*	**label:*support** will show items with a label that ends with "support."

Legal

© 2023 GitHub, Inc.   <u>Terms</u>   <u>Privacy</u>   <u>Status</u>   <u>Pricing</u>   <u>Expert services</u>   <u>Blog</u>