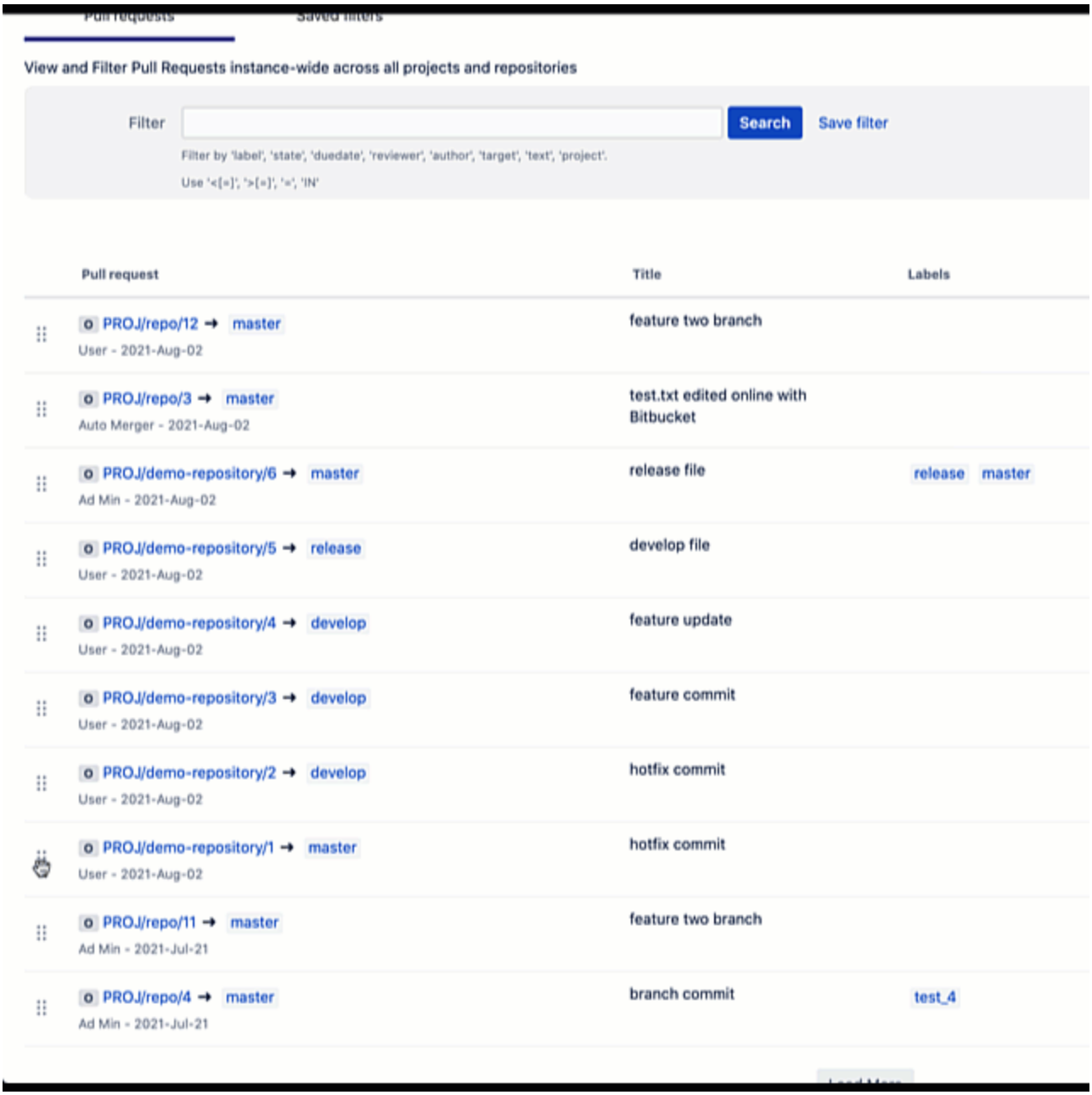
Pull request merge structure

Feature Guide Organizr · https://izymes.com/docs/pull-request-merge-structure/

Organizr structured merge hierarchy enables your team to prioritize and structure pull requests across repositories, projects and the entire Bitbucket instance.

Ensure pull requests are merged in order of priority. Structure the sequence in which multiple pull requests should be merged.

Dependent pull requests need to be merged before the current one can be merged.



Linking and unlinking pull request dependencies

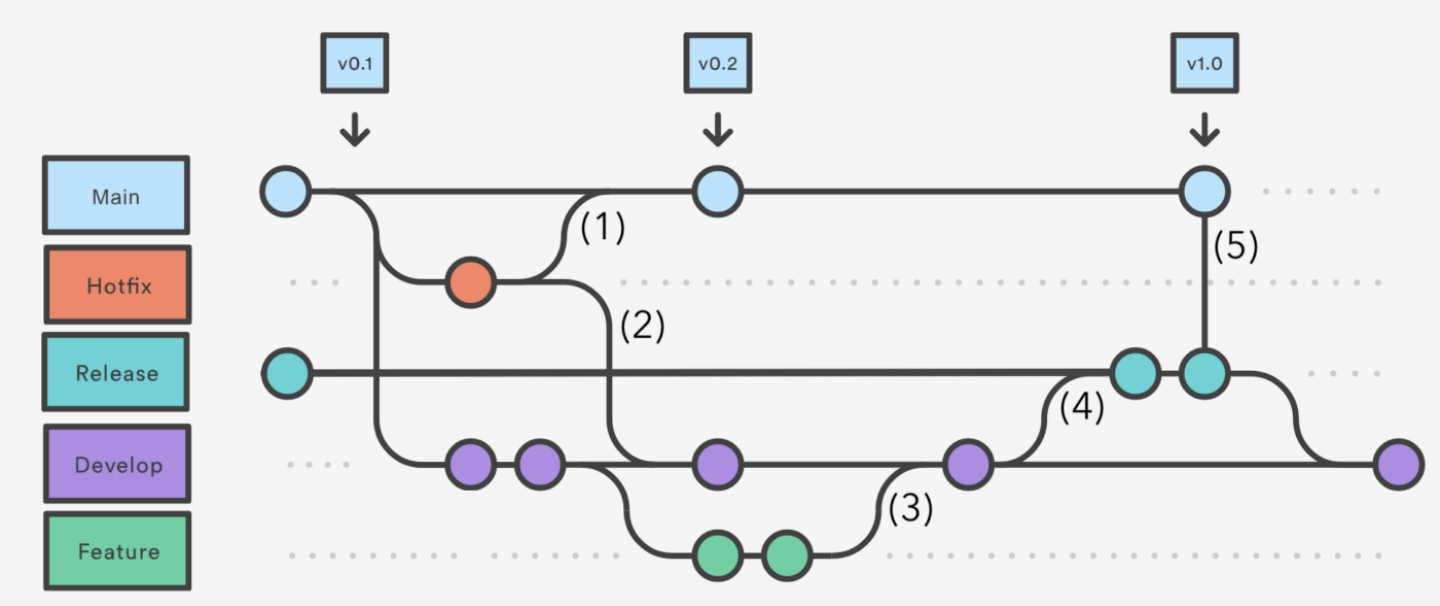
When it comes to getting the next release out and deployed – more often than not things don't go as smooth as you wish. There's a last-minute bugfix and another feature just got finished, plus there is a bug in production that needs to be fixed immediately.

If your team is using a source code branching model that's close to Git flow – all of the above should not be a problem.

As always, prioritizing pull request merges to ship the hotfix, the bugfix and perhaps even the new feature in the upcoming release is key to stay on top of the release process.

For example, if you make a pull request from develop to release branches dependent on a pull request from feature to develop branches then the feature branch must be merged first before the develop branch can be merged.

With **Organizr pull request merge hierarchy** release managers and their team can make sure pull requests are merged in order of priority.



(1) Top priority – that hotfix needs to go into production

(2) The hotfix also needs to be merged with develop to avoid drift between develop and master

(3) That feature the team's been working on is ready to be merged to develop

(4) Feature and bugfix can be merged to release for QA testing

(5) The feature can is shipped to production

Here is a short clip that shows Organizr pull request merge hierarchies in action.

Embedded media: http://www.izymes.com/wp-content/uploads/2021/08/Converted-pr-dependencies-2021.mp4?_=1

Try Organizr!