Using Reviewable for Large Pull Requests

Bogged Down by the Code Review Process?

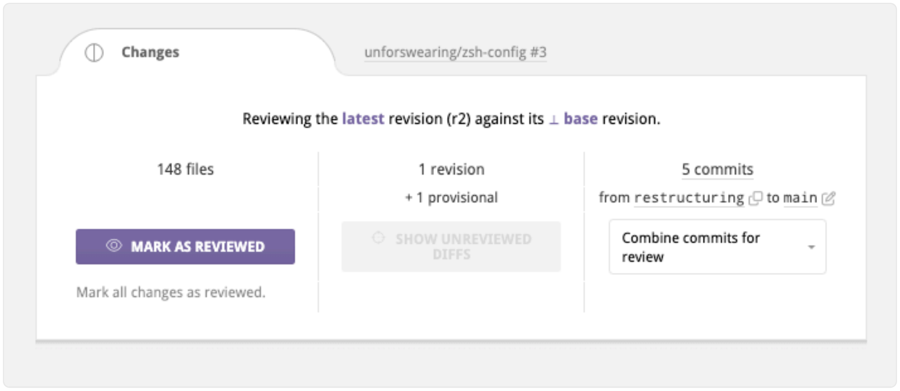
By: Alvin Charity

Published: Friday, June 14, 2024

Code review is an important part of any developer's process, and is essential to helping your team create the highest quality code possible. Ideally, code reviews would be limited to the scope of a feature implementation or bug fix that can be quickly reviewed, letting your developers get back to coding. With over 100 million developers, it is likely that your code lives on Github, and their lightweight code review tools are great for these small, self-contained pull requests. However, large teams may find that code reviews consist of hundreds of files that contain thousands of lines that need to be modified at the same time to ensure the code continues to function together. Unfortunately, despite being the most popular code hosting and project management platform, Github is not the best at handling large files. Understandably, Github imposes some limits on the number of files that can be included in PR diffs. In some instances, these limits may cause your browser to crash when attempting to review a PR that contains 50 or more files.

How Reviewable helps manage large pull requests

Reviewable allows you to create custom completion conditions that lets you control many aspects of your code review process. For example, Reviewable also automatically skips vendored files, meaning you no longer have to worry about scanning through code added to your repository via your language's package manager, or files dumped into your repository by coding tools. These features allow you to reduce friction in your team, speed up your review process, and never worry about having too many files in your PR. For example, I maintain a complex shell environment, and currently have a pull request that contains 148 files. Some of these files will need a thorough review, but others can safely be ignored. Manually reviewing this number of files may take a while, and can be a very tedious process.


Changes
unforswearing/zsh-config #3

Reviewing the latest revision (r2) against its ⊥ base revision.

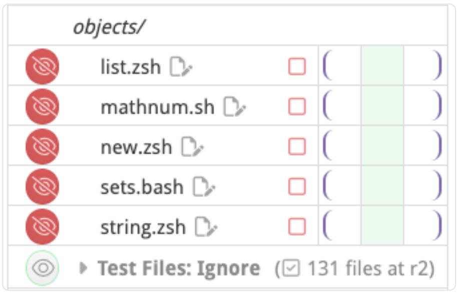
148 files | 1 revision + 1 provisional | 5 commits from restructuring to main

MARK AS REVIEWED
Mark all changes as reviewed.
SHOW UNREVIEWED DIFFS
Combine commits for review

Reviewable allows you to use custom completion conditions to reduce some of your workload. These completion conditions can be edited in the repository settings section of Reviewable, and provides a live evaluation environment where you can see how your rules apply in real-time. Code you add to your completion condition is used to change the state of your review using the `review` object as a starting point. This object provides access to the current pull request, including files, discussions, and other information.

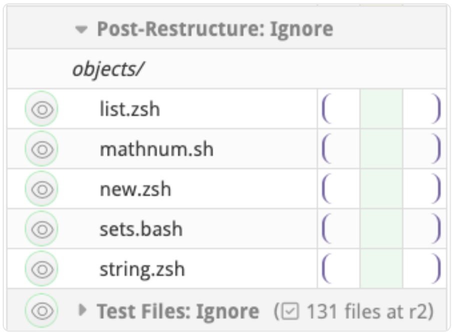
Managing Files Using Completion Conditions

I mentioned above that some of these files can safely be ignored. In my case this includes any files in the `/objects` directory. Since this code has been created in preparation for an update in the future, we want to review these files automatically to get them out of the way.



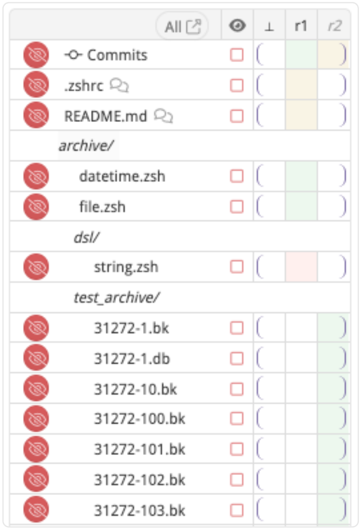
objects/
list.zsh
mathnum.sh
new.zsh
sets.bash
string.zsh
Test Files: Ignore (131 files at r2)

Now these files have been reviewed and are grouped to indicate that no additional review is required.



Post-Restructure: Ignore
objects/
list.zsh
mathnum.sh
new.zsh
sets.bash
string.zsh
Test Files: Ignore (131 files at r2)

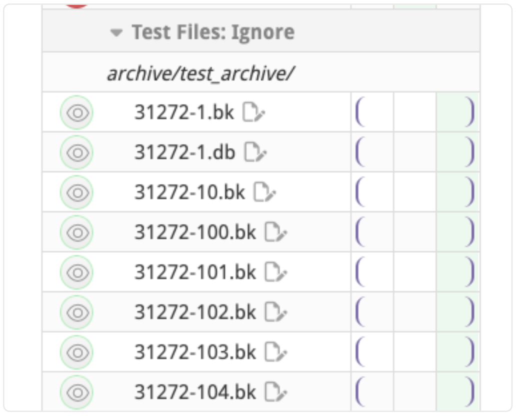
After further inspection, it seems that most of the files that are taking up space in my repository are temporary files from external programs, located in the `/test` and `/archive/test_archive` directories.



All ⊥ r1 r2
Commits
.zshrc
README.md
archive/
datetime.zsh
file.zsh
dsl/
string.zsh
test_archive/
31272-1.bk
31272-1.db
31272-10.bk
31272-100.bk
31272-101.bk
31272-102.bk
31272-103.bk

Since these can also be safely ignored, we can apply a similar completion condition to mark these files as reviewed, and add these files to a group on the Reviewable dashboard.

Now all of the files in the `/test` and `/archive/test_archive` directories have been marked as reviewed and are grouped under the heading "Test Files: Ignore".

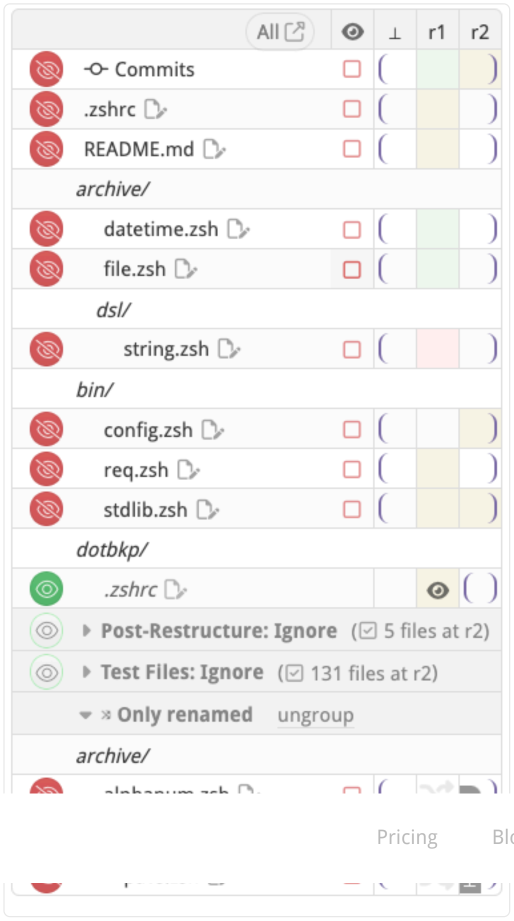


Test Files: Ignore

archive/test_archive/

31272-1.bk
31272-1.db
31272-10.bk
31272-100.bk
31272-101.bk
31272-102.bk
31272-103.bk
31272-104.bk

To keep these grouped files out of the way, we can collapse the `Post-Restructure: Ignore` and `Tests Files: Ignore` groups.



	All		⊥	r1	r2
	Commits				
	.zshrc				
	README.md				
	archive/				
	datetime.zsh				
	file.zsh				
	dsl/				
	string.zsh				
	bin/				
	config.zsh				
	req.zsh				
	stdlib.zsh				
	dotbkp/				
	.zshrc				
	Post-Restructure: Ignore (5 files at r2)				
	Test Files: Ignore (131 files at r2)				
	Only renamed ungroup				
	archive/				

And here is the code for the completion condition.

```
// dependencies: lodash4
_.forEach(review.files, (file) => {

  // This example will automatically review files in the `/ob-
jects` directory
  // and group them into a new section of the dashboard.
  if (/objects/i.test(file.path)) {
    _.forEach(file.revisions, (rev) => {
      rev.reviewed = true;
    });
    file.group = "Post-Restructure: Ignore";
  }

  // This example marks all files in the `/test` and
`/archive/test_archive` directories
  //  as reviewed and groups the test files into a new section of
the dashboard.
  if (/test/i.test(file.path)) {
    _.forEach(file.revisions, (rev) => {
      rev.reviewed = true;
      file.reviewers = [{ username: "unforswearing" }];
    });
    file.group = "Test Files: Ignore";
  }
});

// Return the review object with updated files.
return {
  files: review.files,
};
```

With just a few lines of code I was able to automatically review and group 136 files in this pull request, leaving only 12 files to review, saving time and reducing friction. Check out the documentation for completion conditions or take a look at some example completion conditions to help get you started. Reviewable helps your team manage code review on your terms, without having to sacrifice developer time, or settle for slow or ineffective tools. Github can get you started, let Reviewable take you further.