(Obsolete) HowToContribute

** The contents have been moved to https://github.com/apache/lucene/blob/main/CONTRIBUTING.md **

How to Contribute to Lucene

Working With Code

Getting the source code

First of all, you need the Lucene source code.

Get the source code using: `git clone https://github.com/apache/lucene`

Notes for required Java version

Be sure that you are using an appropriate version of the JDK. Current main branch requires Java 11.

Unit Tests

Please make sure that precommit and all unit tests succeed before constructing your patch: `./gradlew test`.

To run a single test case from the `lucene/core` directory in your working copy: `./gradlew -p lucene/core test --tests NameOfYourUnitTest`. Run `./gradlew helpTests` to get more information about running tests.

In case your contribution fixes a bug, please create a new test case that fails before your fix, to show the presence of the bug and ensure it never re-occurs. A test case showing the presence of a bug is also a good contribution by itself.

gradle precommit

Please run `./gradlew precommit` to identify problems that should be addressed before you contribute your changes.

Contributing your work

You have two options to contribute your work: you can either create a patch and attach it to an issue on Jira, or open a pull request at `https://github.com/apache/lucene` - whichever works best for you.

Please be patient. Committers are busy people too. If no one responds to your patch after a few days, please make friendly reminders. Please incorporate others' suggestions into into your patch if you think they're reasonable. Finally, remember that even a patch that is not committed is useful to the community.

Opening a pull request

Please refer to GitHub's documentation for an explanation of how to create a pull request. The pull request title should start with the Jira issue number that your work is related to, this way your pull request will get automatically linked from the Jira issue.

Creating a patch

Note that you do not need to create a patch if you already opened a pull request on GitHub.

Before creating your patch, you may want to get 'main' up to date with the latest from upstream. This will help avoid the possibility of others finding merge conflicts when applying your patch. This can be done with `git pull` if main is the current branch.

If your changes are in your git working tree (i.e. not committed to a branch), then do this:

1. Check to see what files you have modified with: `git status`
2. Add any new files with: `git add src/.../MyNewClass.java`
3. In order to create a patch, just type `git diff HEAD > LUCENE-NNNN.patch`

If your changes are all committed to the current (feature) branch:

1. `git diff main... > LUCENE-NNNN.patch`

If some of your changes are committed to the current (feature) branch but you have working-copy changes too:

1. `git diff {{git merge-base main head` > LUCENE-NNNN.patch}}

This will save a diff into the `LUCENE-NNNN.patch` file. Read the patch file. Make sure it includes ONLY the modifications required to fix a single issue.

Patches should be attached to an issue in [Jira](#)

To enable automatic validation of Jira patches, switch the issue's status to "Patch Available" by clicking on the "Enable Patch Review" button near the top of the page. This will enqueue an ASF Jenkins job [PreCommit-LUCENE-Build](#) to run various quality checks on the patch and post a validation report as a comment (by "Lucene/Solr QA") on the issue. Expect a delay of 12 hours or so before the patch validation job actually runs. Note that in order for a patch file to trigger validation, its name must conform to the naming rules outlined here: https://yetus.apache.org/documentation/in-progress/precommit-patchnames/

Stay involved

Contributors should join the [Lucene mailing lists](#). In particular, the commit list (to see changes as they are made), the dev list (to join discussions of changes) and the user list (to help others).

Please keep discussions about Lucene on list so that everyone benefits. Emailing individual committers with questions about specific Lucene issues is discouraged. See http://people.apache.org/~hossman/#private_q.

Getting your feet wet: where to begin?

New to Lucene? Want to find JIRA issues that you can work on without taking on the whole world?

The Lucene/Solr developers use the "newdev" label to mark issues that developers new to Lucene might be interested in working on. The rough criteria used to make this selection are:

- Nobody has done any work on the issue yet.
- The issue is likely not controversial.
- The issue is likely self-contained with limited scope.

To see a list of open Lucene and Solr issues with the newdev label, look at this link http://s.apache.org/newdevlucenesolr

Note: Fixing these issues may require asking questions on the developer list to figure out what they mean - there is no guarantee that any of these will be either quick or easy.

Developer tips

For more contribution guidelines and tips, see [DeveloperTips](#)