<!--

~ Licensed to the Apache Software Foundation (ASF) under one

~ or more contributor license agreements. See the NOTICE file

~ distributed with this work for additional information

~ regarding copyright ownership. The ASF licenses this file

~ to you under the Apache License, Version 2.0 (the

~ "License"); you may not use this file except in compliance

~ with the License. You may obtain a copy of the License at

~

~ http://www.apache.org/licenses/LICENSE-2.0

~

~ Unless required by applicable law or agreed to in writing,

~ software distributed under the License is distributed on an

~ "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY

~ KIND, either express or implied. See the License for the

~ specific language governing permissions and limitations

~ under the License.

-->

# How to Contribute

When submitting a pull request (PR), please use the following guidelines:

- Make sure your code respects existing formatting conventions. In general, follow

the same coding style as the code that you are modifying.

- For Intellij you can import our code style settings xml: [`druid\_intellij\_formatting.xml`](

https://github.com/apache/druid/raw/master/dev/druid\_intellij\_formatting.xml).

- For Eclipse you can import our code style settings xml: [`eclipse\_formatting.xml`](

https://github.com/apache/druid/raw/master/dev/eclipse\_formatting.xml).

- Do add/update documentation appropriately for the change you are making.

- If you are introducing a new feature you may want to first write about your idea

for feedback to [dev@druid.apache.org](https://lists.apache.org/list.html?dev@druid.apache.org). Or create an issue

using "Feature/Change" template. Non-trivial features should include unit tests covering the new functionality. Open

a "Proposal" issue for large changes.

- Bugfixes should include a unit test or integration test reproducing the issue.

- Do not use author tags/information in the code.

- Try to keep pull requests short and submit separate ones for unrelated

features, but feel free to combine simple bugfixes/tests into one pull request.

- If you are adding or updating a dependency, be sure to update the version, license, or notice information in

[licenses.yaml](https://github.com/apache/druid/blob/master/licenses.yaml) as appropriate to help ease

LICENSE and NOTICE management for ASF releases.

You can find more developers' resources in [`dev/`](dev) directory.

## GitHub Workflow

1. Fork the apache/druid repository into your GitHub account

https://github.com/apache/druid/fork

2. Clone your fork of the GitHub repository

```sh

git clone git@github.com:<username>/druid.git

```

replace `<username>` with your GitHub username.

3. Add a remote to keep up with upstream changes

```

git remote add upstream https://github.com/apache/druid.git

```

If you already have a copy, fetch upstream changes

```

git fetch upstream master

```

4. Create a feature branch to work in

```

git checkout -b feature-xxx remotes/upstream/master

```

5. \_Before submitting a pull request\_ periodically rebase your changes

(but don't do it when a pull request is already submitted)

```

git pull --rebase upstream master

```

6. Before submitting a pull request, combine ("squash") related commits into a single one

```

git rebase -i upstream/master

```

This will open your editor and allow you to re-order commits and merge them:

- Re-order the lines to change commit order (to the extent possible without creating conflicts)

- Prefix commits using `s` (squash) or `f` (fixup) to merge extraneous commits.

7. Submit a pull-request

```

git push origin feature-xxx

```

Go to your Druid fork main page

```

https://github.com/<username>/druid

```

If you recently pushed your changes GitHub will automatically pop up a

`Compare & pull request` button for any branches you recently pushed to. If you

click that button it will automatically offer you to submit your pull-request

to the apache/druid repository.

- Give your pull-request a meaningful title.

- In the description, explain your changes and the problem they are solving.

8. Addressing code review comments

Address code review comments by committing changes and pushing them to your feature

branch.

```

git push origin feature-xxx

```

### If your pull request shows conflicts with master

If your pull request shows conflicts with master, merge master into your feature branch:

```

git merge upstream/master

```

and resolve the conflicts. After resolving conflicts, push your branch again:

```

git push origin feature-xxx

```

\_Avoid rebasing and force pushes after submitting a pull request,\_ since these make it

difficult for reviewers to see what you've changed in response to their reviews. The Druid

committer that merges your change will rebase and squash it into a single commit before

committing it to master.

## FAQ

### Help! I merged changes from upstream and cannot figure out how to resolve conflicts when rebasing!

Never fear! If you occasionally merged upstream/master, here is another way to squash your changes into a single commit:

1. First, rename your existing branch to something else, e.g. `feature-xxx-unclean`

```

git branch -m feature-xxx-unclean

```

2. Checkout a new branch with the original name `feature-xxx` from upstream. This branch will supercede our old one.

```

git checkout -b feature-xxx upstream/master

```

3. Then merge your changes in your original feature branch `feature-xxx-unclean` and create a single commit.

```

git merge --squash feature-xxx-unclean

git commit

```

4. You can now submit this new branch and create or replace your existing pull request.

```

git push origin [--force] feature-xxx:feature-xxx

```