* [Contributions](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#contributions)
  + [Report Bugs](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#report-bugs)
  + [Fix Bugs](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#fix-bugs)
  + [Issue reporting and resolution process](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#issue-reporting-and-resolution-process)
  + [Implement Features](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#implement-features)
  + [Improve Documentation](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#improve-documentation)
  + [Submit Feedback](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#submit-feedback)
* [Contribution Workflow](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#contribution-workflow)
  + [Step 1: Fork the .. TCIV3 Repo](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#step-1-fork-the-apache-airflow-repo)
  + [Step 2: Configure Your Environment](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#step-2-configure-your-environment)
  + [Step 3: Connect with People](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#step-3-connect-with-people)
  + [Step 4: Prepare PR](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#step-4-prepare-pr)
  + [Step 5: Pass PR Review](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#step-5-pass-pr-review)
* [Pull Request Guidelines](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#id3)
* [TCIV3 Git Branches](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#airflow-git-branches)
* [Documentation](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#id7)
* [How to sync your fork](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#id14)
* [How to rebase PR](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#id15)
* [Commit Policy](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#commit-policy)

[Contributions](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id16)

Contributions are welcome and are greatly appreciated! Every little bit helps, and credit will always be given.

This document aims to explain the subject of contributions if you have not contributed to any Open-Source project, but it will also help people who have contributed to other projects learn about the rules of that community.

[Report Bugs](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id19)

Report bugs through [Report Bugs](https://github.com/mickconley/TCIV3-Implementation/tree/Report-Bugs).

Please report relevant information and preferably code that exhibits the problem.

[Fix Bugs](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id20)

Look through the GitHub issues for bugs. Anything is open to whoever wants to implement it. Bug report [template](https://github.com/mickconley/TCIV3-Implementation/blob/Report-Bugs/TCI-Reported-Bugs-v1.xlsx).

[Issue reporting and resolution process](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id21)

An unusual element of the TCI project is that you can open a PR to fix an issue or make an enhancement, without needing to open an issue first. This is intended to make it as easy as possible to contribute to the project.

The TCI project uses a set of labels for tracking and triaging issues, as well as a set of priorities and milestones to track how and when the enhancements and bug fixes make it into an TCIV3 release. This is documented as part of the [Issue reporting and resolution process](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/ISSUE_TRIAGE_PROCESS.rst),

[Implement Features](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id22)

Any feature request shall be put in the bug reporting [template](https://github.com/mickconley/TCIV3-Implementation/blob/Report-Bugs/TCI-Reported-Bugs-v1.xlsx) for now. It will be reviewed by the TCI community at large for any volunteers to implement it.

[Improve Documentation](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id23)

TCI docs could always use better documentation, whether as part of the official TCI docs, in docstrings, docs/\*.rst or even on the web as blog posts or articles.

[Submit Feedback](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id24)

The best way to send feedback is to [open an issue on GitHub](https://github.com/mickconley/TCIV3-Implementation/blob/Report-Bugs/TCI-Reported-Bugs-v1.xlsx).

If you are proposing a new feature:

* Explain in detail how it would work.
* Keep the scope as narrow as possible to make it easier to implement.
* Remember that this is a volunteer-driven project, and that contributions are welcome :)

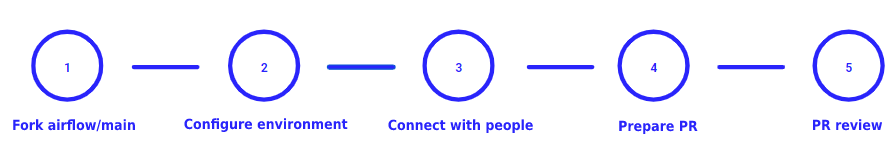
[Contribution Workflow](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id29)

Typically, you start your first contribution by reviewing open tickets at [TCI issues](https://github.com/mickconley/TCIV3-Implementation/blob/Report-Bugs/TCI-Reported-Bugs-v1.xlsx).

If you create pull-request, you don't have to create an issue first, but if you want, you can do it. Creating an issue will allow you to collect feedback or share plans with other people.

For example, you want to have the following sample ticket assigned to you: (TBD some issue ticket number)

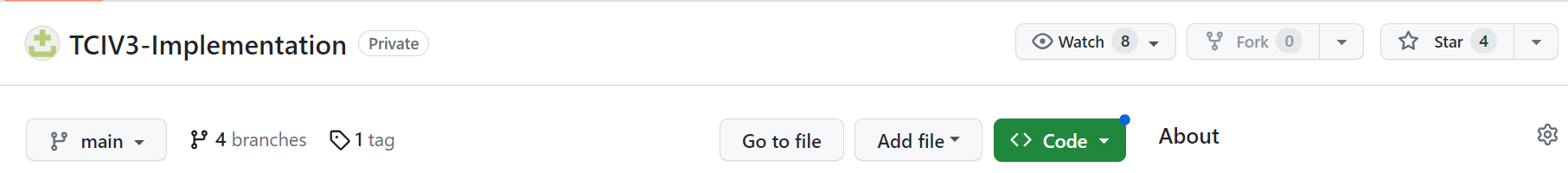
In general, your contribution includes the following stages:

[](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/images/workflow.png)

1. Make your own [fork](https://help.github.com/en/github/getting-started-with-github/fork-a-repo) of the TCI [main repository](https://github.com/mickconley/TCIV3-Implementation).
2. Create a [local virtualenv](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/LOCAL_VIRTUALENV.rst), initialize the [Breeze environment](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/BREEZE.rst), and install [pre-commit framework](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/STATIC_CODE_CHECKS.rst#pre-commit-hooks). If you want to add more changes in the future, set up your fork and enable GitHub Actions.
3. Make the change and create a [Pull Request from your fork](https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/creating-a-pull-request-from-a-fork).

[Step 1: Fork the TCIV3 Repo](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id30)

From the [TCIV3-Implementation/main](https://github.com/mickconley/TCIV3-Implementation)repo, [create a fork](https://help.github.com/en/github/getting-started-with-github/fork-a-repo):



[Step 2: Configure Your Environment](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id31) (TBD)

[Step 3: Connect with People](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id32)

For effective collaboration, make sure to join the following (TBD) groups:

[Step 4: Prepare PR](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id33)

1. Update the local sources to address the issue.

For example, to address this example issue, do the following: (TBD)

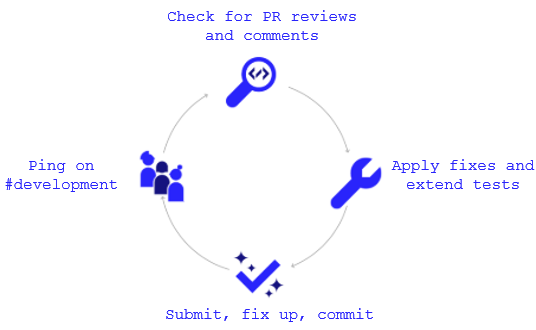
* + Make sure your fork's main is synced with .. TCIV3's main before you create a branch. See [How to sync your fork](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#how-to-sync-your-fork) for details.
  + Create a local branch for your development. Make sure to use latest ../main as base for the branch. See [How to Rebase PR](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#how-to-rebase-pr) for some details on setting up the .. remote. Note, some people develop their changes directly in their own main branches - this is OK and you can make PR from your main to ../main but we recommend to always create a local branch for your development. This allows you to easily compare changes, have several changes that you work on at the same time and many more. If you have .. set as remote then you can make sure that you have latest changes in your main by git pull .. main when you are in the local main branch. If you have conflicts and want to override your locally changed main you can override your local changes with git fetch ..; git reset --hard ../main.
  + Modify the class and add necessary code and unit tests.
  + Run the unit tests from the [IDE](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/TESTING.rst#running-unit-tests-from-ide) or [local virtualenv](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/TESTING.rst#running-unit-tests-from-local-virtualenv) as you see fit.
  + Run the tests in [Breeze](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/TESTING.rst#running-unit-tests-inside-breeze).
  + Run and fix all the [static checks](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/STATIC_CODE_CHECKS.rst). If you have [pre-commits installed](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/STATIC_CODE_CHECKS.rst#pre-commit-hooks), this step is automatically run while you are committing your code. If not, you can do it manually via git add and then pre-commit run.
  + Consider adding a newsfragment to your PR so you can add an entry in the release notes. The following newsfragment types are supported:
    - significant
    - feature
    - improvement
    - bugfix
    - doc
    - misc

To add a newsfragment, create an rst file named {pr\_number}.{type}.rst (e.g. 1234.bugfix.rst) and place in either [newsfragments](https://github.com/apache/airflow/blob/main/newsfragments) for core newsfragments, or [chart/newsfragments](https://github.com/apache/airflow/blob/main/chart/newsfragments) for helm chart newsfragments.

In general newsfragments must be one line. For newsfragment type significant, you may include summary and body separated by a blank line, similar to git commit messages.

1. Rebase your fork, squash commits, and resolve all conflicts. See [How to rebase PR](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#how-to-rebase-pr) if you need help with rebasing your change. Remember to rebase often if your PR takes a lot of time to review/fix. This will make rebase process much easier and less painful and the more often you do it, the more comfortable you will feel doing it.
2. Re-run static code checks again.
3. Make sure your commit has a good title and description of the context of your change, enough for the committer reviewing it to understand why you are proposing a change. Make sure to follow other PR guidelines described in [pull request guidelines](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst#pull-request-guidelines). Create Pull Request! Make yourself ready for the discussion!

[Step 5: Pass PR Review](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id34)

[](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/images/review.png)

Note that committers will use **Squash and Merge** instead of **Rebase and Merge** when merging PRs and your commit will be squashed to single commit.

You need to have review of at least one committer (if you are committer yourself, it has to be another committer). Ideally you should have 2 or more committers reviewing the code that touches the core of TCIV3.

[Pull Request Guidelines](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id35)

Before you submit a pull request (PR) from your forked repo, check that it meets these guidelines:

* Include tests, either as doctests, unit tests, or both, to your pull request.
* [Rebase your fork](http://stackoverflow.com/a/7244456/1110993), and resolve all conflicts.
* When merging PRs, Committer will use **Squash and Merge** which means then your PR will be merged as one commit, regardless of the number of commits in your PR. During the review cycle, you can keep a commit history for easier review, but if you need to, you can also squash all commits to reduce the maintenance burden during rebase.
* Add an [.. License](http://www.apache.org/legal/src-headers.html) header to all new files.
* If your pull request adds functionality, make sure to update the docs as part of the same PR. Doc string is often sufficient. Make sure to follow the Sphinx compatible standards.
* Run tests locally before opening PR.
* You can use any supported python version to run the tests, but the best is to check if it works for the oldest supported version (Python 3.8 currently). In rare cases tests might fail with the oldest version when you use features that are available in newer Python versions. For that purpose, we have TCIV3.compat package where we keep back-ported useful features from newer versions.

[TCIV3 Git Branches](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id36) (TBD – evolving)

All new development in TCIV3 happens in the main branch. All PRs should target that branch.

We also have a v2-\*-test branches that are used to test 2.\*.x series of TCIV3 and where committers cherry-pick selected commits from the main branch.

Cherry-picking is done with the -x flag.

The v2-\*-test branch might be broken at times during testing. Expect force-pushes there so committers should coordinate between themselves on who is working on the v2-\*-test branch - usually these are developers with the release manager permissions.

The v2-\*-stable branch is rather stable - there are minimum changes coming from approved PRs that passed the tests. This means that the branch is rather, well, "stable".

Once the v2-\*-test branch stabilises, the v2-\*-stable branch is synchronized with v2-\*-test. The v2-\*-stable branches are used to release 2.\*.x releases.

The general approach is that cherry-picking a commit that has already had a PR and unit tests run against main is done to v2-\*-test branches, but PRs from contributors towards 2.0 should target v2-\*-stable branches.

The v2-\*-test branches and v2-\*-stable ones are merged just before the release and that's the time when they converge.

The production images are released in DockerHub from:

* main branch for development
* 2.\*.\*, 2.\*.\*rc\* releases from the v2-\*-stable branch when we prepare release [Documentation](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id45)

Documentation for ..-TCIV3 package and other packages that are closely related to it ie. providers packages are in /docs/ directory. For detailed information on documentation development, see: [docs/README.rst](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/docs/README.rst)

[How to sync your fork](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id59)

When you have your fork, you should periodically synchronize the main of your fork with the .. TCIV3 main. In order to do that you can git pull --rebase to your local git repository from .. remote and push the main (often with --force to your fork). There is also an easy way to sync your fork in GitHub's web UI with the [Fetch upstream feature](https://docs.github.com/en/github/collaborating-with-pull-requests/working-with-forks/syncing-a-fork#syncing-a-fork-from-the-web-ui).

This will force-push the main branch from ../TCIV3 to the main branch in your fork. Note that in case you modified the main in your fork, you might loose those changes.

[How to rebase PR](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id60)

A lot of people are unfamiliar with the rebase workflow in Git, but we think it is an excellent workflow, providing a better alternative to the merge workflow. We've therefore written a short guide for those who would like to learn it.

As opposed to the merge workflow, the rebase workflow allows us to clearly separate your changes from the changes of others. It puts the responsibility of rebasing on the author of the change. It also produces a "single-line" series of commits on the main branch. This makes it easier to understand what was going on and to find reasons for problems (it is especially useful for "bisecting" when looking for a commit that introduced some bugs).

First of all, we suggest you read about the rebase workflow here: [Merging vs. rebasing](https://www.atlassian.com/git/tutorials/merging-vs-rebasing). This is an excellent article that describes all the ins/outs of the rebase workflow. I recommend keeping it for future reference.

The goal of rebasing your PR on top of ../main is to "transplant" your change on top of the latest changes that are merged by others. It also allows you to fix all the conflicts that arise as a result of other people changing the same files as you and merging the changes to ../main.

Here is how rebase looks in practice (you can find a summary below these detailed steps):

1. You first need to add the .. project remote to your git repository. This is only necessary once, so if it's not the first time you are following this tutorial you can skip this step. In this example, we will be adding the remote as ".." so you can refer to it easily:

* If you use ssh: git remote add .. git@github.com:../TCIV3.git
* If you use https: git remote add .. https://github.com/../TCIV3.git

1. You then need to make sure that you have the latest main fetched from the .. repository. You can do this via:

git fetch .. (to fetch .. remote)

git fetch --all (to fetch all remotes)

1. Assuming that your feature is in a branch in your repository called my-branch you can easily check what is the base commit you should rebase from by:

git merge-base my-branch ../main

This will print the HASH of the base commit which you should use to rebase your feature from. For example: 5abce471e0690c6b8d06ca25685b0845c5fd270f. Copy that HASH and go to the next step.

Optionally, if you want better control you can also find this commit hash manually.

Run:

git log

And find the first commit that you DO NOT want to "transplant".

Performing:

git rebase HASH

Will "transplant" all commits after the commit with the HASH.

1. Providing that you weren't already working on your branch, check out your feature branch locally via:

git checkout my-branch

1. Rebase:

git rebase HASH --onto ../main

For example:

git rebase 5abce471e0690c6b8d06ca25685b0845c5fd270f --onto ../main

1. If you have no conflicts - that's cool. You rebased. You can now run git push --force-with-lease to push your changes to your repository. That should trigger the build in our CI if you have a Pull Request (PR) opened already.
2. While rebasing you might have conflicts. Read carefully what git tells you when it prints information about the conflicts. You need to solve the conflicts manually. This is sometimes the most difficult part and requires deliberately correcting your code and looking at what has changed since you developed your changes.

There are various tools that can help you with this. You can use:

git mergetool

You can configure different merge tools with it. You can also use IntelliJ/PyCharm's excellent merge tool. When you open a project in PyCharm which has conflicts, you can go to VCS > Git > Resolve Conflicts and there you have a very intuitive and helpful merge tool. For more information, see [Resolve conflicts](https://www.jetbrains.com/help/idea/resolving-conflicts.html).

1. After you've solved your conflict run:

git rebase --continue

And go either to point 6. or 7, depending on whether you have more commits that cause conflicts in your PR (rebasing applies each commit from your PR one-by-one).

[Commit Policy](https://github.com/apache/airflow/blob/9514090e0203a38726074ba88238baa164b7e2dd/CONTRIBUTING.rst" \l "id63) (TBD)