How to Contribute

Apache Way

ASF: How It Works

ASF: The Apache Way

Apache Way, Apache Con (Slides)

Introduction

Please start from sending out an introduction email to the project's dev mailing list (at first: subscribe) with a request for contributors permissions.

First email to the dev mailing list (example)

To: dev@ignite.apache.org

Subject: I want to contribute to Apache ignite

Hello, Everyone!

My name is [Name]. I want to contribute to Ignite. I would like to start from [identification of particular ticket or module, if you have one in mind]. Please, help me to start contributing.

My ASF JIRA username is [username]*.

Regards,

[Name]

*learn more about ASF JIRA below

JIRA Process

JIRA Accounts

If you do not have ASF JIRA account, use https://selfserve.apache.org/jira-account.html to request a new one before going to the next step.

Once you have an ASF JIRA account, send out an introduction email to the project's dev mailing list (at first: subscribe) with a request for contributors permissions. PMCs should handle this request and grant contributor access to a new community member.

Getting Started

- Easy tickets to get started with Apache Ignite: open tickets with `newbie` label
- All other issues: Ignite Jira

IMPORTANT

The JIRA handling process outlined below should be followed in absolutely all cases, without exceptions, regardless of the ticket complexity.

Tickets and Versions

Tickets are picked up by community members from a pool of unassigned and unscheduled tickets.

JIRA issues are grouped by `fixVersion` field. Tickets should be assigned to an unreleased version if they are blockers or regressions for that version. The contributor should make sure that `fixVersion` is set to a proper version that actually releases the changes made in the scope of the ticket. Open and in-progress tickets may have this field blank.

The following page contains information on upcoming releases: Release Planning.

Ticket Creation

- Every JIRA ticket should be sufficiently described. Avoid filing tickets with title only and without description.
- If possible, please provide a reproducer or runnable test for Bug tickets.
- If there is a discussion pertaining to the ticket, the ticket should have a link to the dev mailing list.

Working on a ticket

- Anyone in the community may start working on any Unassigned ticket.
- (OPTIONAL) It is recommended to clarify the relevance and correctness of the ticket with the community before doing actual implementation. To do this, you can write an email to the dev mailing list and wait for an answer (please do not forget to subscribe to the list first).
- Before beginning to work on a ticket, you should assign the ticket to yourself.
- Move the ticket to `IN PROGRESS` state.
- If necessary, add comments describing the design decisions or approaches you plan to take.

Documenting a ticket

If the changes implemented under the ticket require changes in user documentation, create a *related* documentation ticket (add "Documentation" to the **Co mponent** field) and provide a reasonable amount of details in the ticket's description. The information provided in the ticket should be sufficient for any contributor to start working on it. If there is no need to change user documentation, uncheck the **Docs Required** flag. The **Docs Required** flag is used to filter out the tickets that require documentation so that our documentation is always up to date.

Review Process and Maintainers

- Ignite employs both Review-Then-Commit processes.
- Please consult to Review Checklist to understand how tickets are reviewed and what rules to follow.

Master Branch

Ignite "master" branch should always be **release-ready**. Please avoid any commits or merges to the "master" branch unless the whole TeamCit y CI suite has passed.

Any change should be reviewed by a contributor and passed to a committer (who may or may not be the same as the main reviewer) for merging.

Submitting for Review

- Attach Pull Request URL to ticket (see instructions at Workflow)
- Add a comment describing what has been implemented.
- Run tests using Run All... TeamCity suite. Validate that all tests have passed using TeamCity bot. It allows checking that no new non-flaky tests have failed, and posting the confirmation ("TC green visa") to the ticket.
- Move ticket to `PATCH AVAILABLE` state.
- If you know a contributor who should review this ticket, you should mention them directly:

Optional: Tips to pass review quickly

Check affected files' git history to find a person most likely able to review changes.

In case it's hard to determine who's able to review by git history use maintainers list presented below.

Add "review request" comment to the Jira Issue starting with a contributor username.

for example: "[~avinogradov], Please review my changes."

This user will be notified and will review your changes and/or help to find another contributor to do a review, then will help you finding a committer to merge the change if needed.

- Otherwise, please open a thread on the dev mailing list describing the change and asking for review.
- If there is no review activity after a week has passed, try picking another reviewer or bumping the dev mailing list thread.

Reviewing a Ticket

- Make sure that your patch satisfies Review Checklist rules.
- Each comment should be started with [~username] to guarantee proper notification.
- The reviewer may suggest improvements. It is recommended to leave a comment on the ticket in addition to PR comments. This helps other contributors to identify that patch may need improvement.
- Patch author may implement those improvements or discuss the best course of action with the reviewer.
- The reviewer should add comment like "looks good to me" (LGTM) once the review successfully finished.
- When the ticket is reviewed, any committer may merge it.

Closing a Ticket

- Once the ticket has passed all the reviews and has no additional comments, the committer should apply the latest patch and push it to the master branch.
- The committer should comment on the ticket stating that the patch has been applied to the master.
- File release notes for this ticket, and unset **Release Notes Required** flag.
- Set fixVersion field to target release, usually the next unreleased version.
- Move ticket to `RESOLVED` state.

GIT workflow

Setting up:

- You need to fork an Apache Ignite mirror on GitHub.
- Make a local copy of your Apache Ignite mirror fork. Your remote origin will refer to 'https://github.com/<your_github_uname>/ignite'.
- You will need to update a local master sometimes (to merge to your development branches sometimes). How to do it:
    - Add remote for Apache Ignite mirror (you need to do it once)

      `git remote add upstream https://github.com/apache/ignite`
    - Each time when you want to update your local master do the following:

```
git pull upstream
git checkout master
```

Contributing:

- Fix / implement JIRA ticket in your fork. Provide Java docs whenever required. If you add a new package make sure that package-info.java file in it is in place with a description. Commit branch to origin (origin = your fork). It's recommended to develop IGNITE-nnn ticket at ignite-nnn branch.
- If your contribution is significant (new functionality, deeply reworked existed functionality API) please describe the contribution in detail, then add an example of the usage to 'ignite-example' and send an email about the contribution to the dev mailing list.
- Create a pull request from the new remote branch in the fork to master of Apache Ignite mirror. Please, start a title of the pull request from 'IGNITE-nnn'. An email about the pull request will be sent to the dev mailing list and the same JIRA comment will be added to the IGNITE-nnn ticket.
- Trigger validation of those test suites that have been affected by your changes on TeamCity:
    - Open "Run All..." test suite, press button named "..." that is located on the left of "Run" button. "Run custom build" dialog will appear;
    - Go to "Changes" tab and choose "pull/<pull-request-number>/head" in "Build branch" dropdown list;
    - Press "Run build" button and monitor tests results.
- Inspect contribution using TeamCity Bot:
    - Paste PR or JIRA ticket number to search box;
    - Press More Show "pull/<pull-request-number>/head" report when test results are ready. You can also trigger build from the bot;
    - Press "Comment JIRA" button to leave TC bot visa in JIRA ticket.

- Once tests are passed, the pull request can be reviewed and merged by a committer. Move a corresponding JIRA ticket to "Patch Available" state by clicking on "Submit Patch" button and let the community know that you're ready for review.

Note: Existing pull requests should be updated instead of the creation of new ones, when possible.

Information for committers

In addition to contributors configuration, committers need to have one more remote repo - for working with Apache Git repo. It can be added like this:

- `git remote add apache https://gitbox.apache.org/repos/asf/ignite.git`

To push any branch at Apache repo use

- `git push apache <branch_name>`

To apply a pull-request it's strongly recommended using ./scripts/apply-pull-request.sh script. Script takes 'pull-request-id' as a parameter and do next:

1. Checks that you don't have any uncommitted changes.
2. Checks you are one master branch and the master branch is up-to-date.
3. Updates local master from Apache git repo.
4. Fetches pull request to a local branch:
    - `git fetch upstream pull/<id>/head:pull-<id>-head`
5. Saves an author and a comment of the last commit at pull-<id>-head.
6. Merges from the new branch to master:
    - `git merge --squash pull-<id>-head`
7. Ask you about a custom comment or using the saved comment.
8. Commit to local master. The script automatically sign-off a commit and add "Fixes #<id>." suffix to comment (It will close the pull request, see https://help.github.com/articles/closing-issues-via-commit-messages/):
    - `git commit --author="<saved_author>" -s -m "<comment> - Fixes #<id>."`

Now, you will have one commit at master with all changes from the pull-request. Changes can be reviewed again. If you accept all changes and want to push it, do next:

- `git push apache master`

Create a Ticket Branch (only for committers)

Whenever working on bigger features, committers can also create 'ready to be reviewed' branch ignite-XXXX, where XXXX is the number of the JIRA ticket.

TeamCity should be forced to run all tests on created branch before review. Once tests are passed, the branch has been reviewed by the module's maintainer.

Created branch name should be attached to a JIRA ticket and the ticket status should be changed on Patch Available.

The branch can be merged to master on successful review by at least one another committer.

The branch should be deleted on branch merged to master or issue canceled. Committers are in charge of deleting their branches.

Appendix A. Components and their maintainers

Component		Maintainers
Ignite Core (the rest of internals not covered below)		Anton Vinogradov Alexey Goncharuk
	PME	Pavel Kovalenko Maxim Muzafarov
	Rebalance	Anton Vinogradov Pavel Kovalenko Maxim Muzafarov
	Affinity	Pavel Kovalenko
	PDS	Alexey Goncharuk Maxim Muzafarov
	Encryption	
	MVCC (obsolete)	Igor Seliverstov
	Transactions	Alexey Scherbakov

Snapshots	Maksim Timonin Maxim Muzafarov
Marshalling (Binary, Optimized, JDK)	
Discovery & Communication SPIs	Alexey Goncharuk
Ignite Compute API	
Ignite Services API	Denis Mekhanikov Vyacheslav Daradur
Ignite SQL & Text Queries & JDBC	Iurii Gerzhedovich
Ignite Continuous Queries	
Machine Learning/Deep Learning (ml, TensorFlow, sub-modules in ml)	Alexey Zinoviev
Build System / Releases	Anton Vinogradov
TeamCity Bot	Dmitry Pavlov Petr Ivanov
Spark Integration	Nikolay Izhikov Alexey Zinoviev
.NET API	Pavel Tupitsyn
C++ API	Igor Sapego
Other thin clients (Python, Node.js, PHP, etc)	Igor Sapego
ODBC	Igor Sapego
JDBC	Iurii Gerzhedovich
Streamers (JMS, Flume, Kafka, etc.)	Saikat Maitra
Docker, Mesos, YARN integration	Ilya Kasnacheev Petr Ivanov
AWS, Google Compute Engine, JClouds integration	
Visor Console	
WebSession & WebSession Filter	

External integrations maintainers