# Contributing to Azkaban

:+1: First off, thanks for taking the time to contribute! :+1:

Please take a moment to read this document if you would like to contribute. Please feel free to

reach out on [the Azkaban Google Group](https://groups.google.com/forum/?fromgroups#!forum/azkaban-dev)

if you have any questions.

## Reporting issues

We [use Github issues](https://github.com/azkaban/azkaban/issues) to track bug reports, feature requests,

and submitting pull requests.

If you find a bug:

Use the GitHub issue search to check whether the bug has already been reported.

If the issue has been fixed, try to reproduce the issue using the latest master branch of the repository.

If the issue still reproduces or has not yet been reported, try to isolate the problem before opening an issue.

Please don't report an issue as a way to ask a question. Use the Google group instead.

## Pull requests

Before embarking on making significant changes, please open an issue and ask first so that you do not risk

duplicating efforts or spending time working on something that may be out of scope.

Please see [the Github documentation](https://help.github.com/articles/about-pull-requests/) for

help on how to create a pull request.

Please give your pull request a clear title and description and note which issue(s) your pull request fixes.

\*Important\*: By submitting a patch, you agree to allow the project owners to license your work

under [the Apache 2.0 License](http://www.apache.org/licenses/LICENSE-2.0).

## IDE setup

We recommend [IntelliJ IDEA](https://www.jetbrains.com/idea/). There is a free community

edition available. Azkaban is a standard [Gradle](https://gradle.org/)

project. You can import it into your IDE using the `build.gradle` file in the root directory. For IntelliJ, choose Open

Project from the Quick Start box or choose Open from the File menu and select the root `build.gradle` file.

## Style guides

Azkaban follows the [Google code style](http://google.github.io/styleguide/). The template file,

`az-intellij-style.xml`, can be found in the root directory. It's based on the

[intellij-java-google-style.xml config file](https://github.com/google/styleguide/blob/75c289f1d33836d1ff4bd94e6c9033673e320b58/intellij-java-google-style.xml) from the google/styleguide project.

Follow [the Intellij's code style help](https://www.jetbrains.com/help/idea/2017.1/code-style.html)

to import and set up the style. Make sure to activate the AzkabanStyle by

"copy the IDE scheme to the current project, using the Copy to Project... command."

Install and enable the intellij's

[save actions plugin](https://github.com/dubreuia/intellij-plugin-save-actions)

to reformat/refactor code automatically:

Please turn on the following options only (all of them are available in release 1.1.0 and above, see https://github.com/dubreuia/intellij-plugin-save-actions/releases):

\* Optimize imports

\* Reformat file

\* Add final modifier to field

\* Add final modifier to local variable or parameter

\* Add this to field access

\* Add class qualifier to static member access outside declaring class only

\* Add missing @Override annotations

\* Add blocks to if/while/for statements

\* Add missing serialVersionUID field for Serializable classes

\* Remove final from private method

\* Remove explicit generic type for diamond

\* Remove unnecessary semicolon

Please refer to save actions plugin (https://github.com/dubreuia/intellij-plugin-save-actions) for implication of each option.

### New line at the end of a file

Configure Intellij to automatically insert a new line at the end of a file:

Preferences ? Editor ? General ? Ensure line feed at file end on Save

### todo style

Add your user name to TODO items.

Use the form:

`

todo username: todo details

`

## Coding conventions

- Automated tests are expected.

Exceptions should be rare.

An example is automated UI testing, until the test framework catches up.

- Define configuration keys in the azkaban.Constants class.

- [Favor composition over inheritance](http://thefinestartist.com/effective-java/16).

- Favor @Singleton notation over the bind method.

See [this example PR](https://github.com/azkaban/azkaban/pull/1428).

- [Favor JSR-330's annotations and Provider interface](https://github.com/google/guice/wiki/JSR330).

e.g. `import javax.inject.Inject;` instead of `import com.google.inject.Inject;`

- Introduce an interface only when needed.

When there is a single implementation, don't introduce an interface. When there is a need for a

second implementation, refactor the code to introduce an interface as needed.

The person making that change in the future will know more (two implementations) than you do now (one implementation)

and they will use that knowledge to make better API choices. Modern IDEs have good support for

such refactoring. There is less IDE support for removing an interface.

This convention only applies to the code internal to this project. It's a good use of interfaces

for public APIs, which are used by the code that the project owners can't change and evolve

easily.

See [this blog post](https://rrees.me/2009/01/31/programming-to-interfaces-anti-pattern/).

- Favor small and focused classes, files, methods over large ones.

It's ok to have many classes, files.

Here is [a rule of 30 suggestion](https://dzone.com/articles/rule-30-%E2%80%93-when-method-class-or).

- Use [Mockito](http://site.mockito.org/) as the mocking framework in unit tests.

It's an exception rather than the rule to create custom mocking classes.

- Use [AssertJ](http://joel-costigliola.github.io/assertj/) as the assertion library in unit tests.

- Use slf4j instead of log4j.

Use the form:

`

private static final Logger logger = LoggerFactory.getLogger($CLASS\_NAME$.class);

`

- Add or update the copyright notice for most files. The year should be the year in which the file was originally created.

- When ignoring a test, provide reasons for ignoring the test.

Use the form:

`@Ignore("reasons")`

# Misc

- See [the dev tips wiki](https://github.com/azkaban/azkaban/wiki/Developer-Tools-and-Tips) for

tips.

- Store images in the [azkaban-images repo](https://github.com/azkaban/azkaban-images)

This is useful to store images needed in the wiki pages for example.