# How to contribute

We definitely welcome your patches and contributions to gRPC! Please read the gRPC

organization's [governance rules](https://github.com/grpc/grpc-community/blob/master/governance.md)

and [contribution guidelines](https://github.com/grpc/grpc-community/blob/master/CONTRIBUTING.md) before proceeding.

If you are new to github, please start by reading [Pull Request howto](https://help.github.com/articles/about-pull-requests/)

## Legal requirements

In order to protect both you and ourselves, you will need to sign the

[Contributor License Agreement](https://identity.linuxfoundation.org/projects/cncf).

## Compiling

See [COMPILING.md](COMPILING.md). Specifically, you'll generally want to set

`skipCodegen=true` so you don't need to deal with the C++ compilation.

## Code style

We follow the [Google Java Style

Guide](https://google.github.io/styleguide/javaguide.html). Our

build automatically will provide warnings for style issues.

[Eclipse](https://raw.githubusercontent.com/google/styleguide/gh-pages/eclipse-java-google-style.xml)

and

[IntelliJ](https://raw.githubusercontent.com/google/styleguide/gh-pages/intellij-java-google-style.xml)

style configurations are commonly useful. For IntelliJ 14, copy the style to

`~/.IdeaIC14/config/codestyles/`, start IntelliJ, go to File > Settings > Code

Style, and set the Scheme to `GoogleStyle`.

## Maintaining clean commit history

We have few conventions for keeping history clean and making code reviews easier

for reviewers:

\* First line of commit messages should be in format of

`package-name: summary of change`

where the summary finishes the sentence: `This commit improves gRPC to \_\_\_\_\_\_\_\_\_\_\_\_.`

for example:

`core,netty,interop-testing: add capacitive duractance to turbo encabulators`

\* Every time you receive a feedback on your pull request, push changes that

address it as a separate one or multiple commits with a descriptive commit

message (try avoid using vauge `addressed pr feedback` type of messages).

Project maintainers are obligated to squash those commits into one when

merging.

## Running tests

### Jetty ALPN setup for IntelliJ

The tests in interop-testing project require jetty-alpn agent running in the background

otherwise they'll fail. Here are instructions on how to setup IntellJ IDEA to enable running

those tests in IDE:

\* Settings -> Build Tools -> Gradle -> Runner -> select Gradle Test Runner

\* View -> Tool Windows -> Gradle -> Edit Run Configuration -> Defaults -> JUnit -> Before lauch -> + -> Run Gradle task, enter the task in the build.gradle that sets the javaagent.

Step 1 must be taken, otherwise by the default JUnit Test Runner running a single test in IDE will trigger all the tests.

## Guidelines for Pull Requests

How to get your contributions merged smoothly and quickly.

- Create \*\*small PRs\*\* that are narrowly focused on \*\*addressing a single concern\*\*. We often times receive PRs that are trying to fix several things at a time, but only one fix is considered acceptable, nothing gets merged and both author's & review's time is wasted. Create more PRs to address different concerns and everyone will be happy.

- For speculative changes, consider opening an issue and discussing it first. If you are suggesting a behavioral or API change, consider starting with a [gRFC proposal](https://github.com/grpc/proposal).

- Provide a good \*\*PR description\*\* as a record of \*\*what\*\* change is being made and \*\*why\*\* it was made. Link to a github issue if it exists.

- Don't fix code style and formatting unless you are already changing that line to address an issue. PRs with irrelevant changes won't be merged. If you do want to fix formatting or style, do that in a separate PR.

- Unless your PR is trivial, you should expect there will be reviewer comments that you'll need to address before merging. We expect you to be reasonably responsive to those comments, otherwise the PR will be closed after 2-3 weeks of inactivity.

- Maintain \*\*clean commit history\*\* and use \*\*meaningful commit messages\*\*. See [maintaining clean commit history](#maintaining-clean-commit-history) for details.

- Keep your PR up to date with upstream/master (if there are merge conflicts, we can't really merge your change).

- \*\*All tests need to be passing\*\* before your change can be merged. We recommend you \*\*run tests locally\*\* before creating your PR to catch breakages early on. Also, `./gradlew build` (`gradlew build` on Windows) \*\*must not introduce any new warnings\*\*.

- Exceptions to the rules can be made if there's a compelling reason for doing so.