# How to contribute

[![Gitter](https://badges.gitter.im/Join%20Chat.svg)](https://gitter.im/go-lang-plugin-org/go-lang-idea-plugin?utm\_source=badge&utm\_medium=badge&utm\_campaign=pr-badge&utm\_content=badge)

## Contents

+ [Reporting errors](#reporting-errors)

+ [Contributing to the code](#contributing-to-the-code)

+ [Submitting test cases](#submitting-test-cases)

+ [Building and running the unit tests](#building-and-running-the-unit-tests)

+ [Configuring and debugging in IntelliJ](#configuring-and-debugging-in-IntelliJ)

+ [Code Style](#code-style)

+ [Delve integration](#delve-integration)

+ [Useful links](#useful-links)

## Reporting errors

Before reporting an error, please read the [FAQ](https://github.com/go-lang-plugin-org/go-lang-idea-plugin/wiki/FAQ)

and search for the issue in the issue tracker. Also, please don't bump, +1

or "me too" issues. Thank you.

The simplest way to contribute to the plugin is to report issues you encounter

in your day to day use.

As a rule of thumb, always keep in mind that we are developers just like you. So,

whenever you are going to report an issue, think of how you'd like to receive issues

for your projects. Also, we are doing this in our spare time, so the more information

we have in the report, the faster we can replicate the problem and get on solving it

rather that just doing a bunch of ping-pong in comments trying to extract the needed

details from you.

This information applies also when you don't have an error that is caught by the

built-in facility but it's something that happens and shouldn't happen (say for

example, a formatting issue).

When filing an issue, please answer these questions:

- What version of Go plugin are you using?

- What version of IDEA are you using?

- What version of Java are you using?

- What did you do?

- What did you expect to see?

- What did you see instead?

Also, every time you can, submit the piece of code that's generating the issue.

As it might be some proprietary code, take some time and write the smallest code

sample that can reproduce it and paste it in the issue (or send it as a link to

[Go Playground](http://play.golang.org/). Screenshots are useful, but, just like

you, we can't copy paste code from screenshots either.

Please ensure that the bug is not reported already, this helps us focusing on

working on bug fixes not triage work.

## Contributing to the code

If you want to contribute to the code, go to GitHub and check out the latest version

and follow the instructions on how to build the plugin from source. After that, you

can start picking some [pending tasks](https://github.com/go-lang-plugin-org/go-lang-idea-plugin/issues) on the issue tracker.

Make sure you look for issues tags with [up for grabs](https://github.com/go-lang-plugin-org/go-lang-idea-plugin/labels/up%20for%20grabs)

as these are some of the easier ones to get started with.

### CLA requirement

Contributing to the plugin requires a signed CLA with JetBrains.

You can view the steps necessary for this at [this page](http://www.jetbrains.org/display/IJOS/Contribute#Contribute-ContributeCode).

### Submitting test cases

Submitting test cases is the next best thing you can do to developing on this

project. In fact, you'll actually develop on it since the test code you are

going to contribute is still code.

Whenever your time or knowledge allows, submitting good test cases for either

for existing issues or for issues that you come across will make a huge difference

in the way we spend time to understand the problems and thus solve them.

### Building and running the unit tests

All code can be checked out from our Github repository in the usual way. That is, clone the repository with HTTPS or SSH:

```

$ git clone https://github.com/go-lang-plugin-org/go-lang-idea-plugin.git

Cloning into 'go-lang-idea-plugin'...

```

On Linux/Mac OS X we use gradle as our build system. Gradle is self-installing. This one command

```

$ ./gradlew buildPlugin

```

compiles everything, runs the tests, and builds the plugins. The output appears in `build/distributions`.

### Configuring and debugging in IntelliJ

1. [Download](http://www.jetbrains.com/idea/) IDEA Community or Ultimate

1. Setup the right version of [Grammar-Kit](https://github.com/JetBrains/Grammar-Kit/releases/download/1.4.1/GrammarKit.zip)

1. Make sure that \*\*UI Designer\*\* and \*\*Gradle\*\* plugins are turned on

1. Checkout plugin repo and open the project

1. Open the copy of go-lang-idea-plugin repository (`File | New | Project from Existing Sources...`) with it. Then import Gradle project. The default project module config should work with a recent IDEA 15 version

1. Git revert changes to the .idea folder because IDEA Gradle import blows it away (https://youtrack.jetbrains.com/issue/IDEA-146295)

1. Open `File -> Project Settings`, go to the SDKs entry, click the `+` and select JDK 1.6

1. Go to the Project entry and make sure that the Project SDK is set to selected SDK

1. Wait until the source files of the SDK are indexed

1. Run or debug the \*\*Go\*\* run configuration

If you use Mac, you may check follow path while looking for SDK:

```

/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/

/Library/Java/JavaVirtualMachines/jdk1.8.0\_25.jdk/Contents/

```

Now you can use the run configurations provided by the plugin source code to

run and play.

There are multiple branches of the plugin which correspond to different versions

of the IntelliJ Platform:

- 141 -> can be used for IDEA 14.1

- 144 -> can be used for IDEA 16

The master branch will follow the current stable release of IDEA (currently IDEA 15).

Going to ``` Run -> Run... ``` will provide you with the following run configurations:

+ `Go` will spawn a new IDEA with the latest version of the plugin enabled.

+ `All tests` will run all the test cases available in the project. Please make

sure that all the test cases pass before committing anything (or making a pull request).

+ `Performance tests` will run all performance test cases available in the project.

+ `Build plugin` will build plugin distribution archive. All artifacts are stored in `gradle/distributions` directory.

You can also have a look at some [useful links](#useful-links) for getting started with

plugin development on IntelliJ IDEA platform.

##### IMPORTANT: MacOS X users note

You might get the following error ```Unsupported major.minor version 52``` in the

logs or the plugin might not work at all.

Check the version of Java your IDE is running on. Since in 99.9% of the cases it will

be Java 6, this means that you compiled the plugin with a different version of Java,

for example Java 8.

To fix the error, please use Java JDK 6 to compile the plugin and everything should work.

To get the log files, you can go to ```Help | Show Log in File Manager``` and then the

log will be displayed to you.

### Code Style

\* Please don't use class comments with information about author or date and time creation.

\* Please don't commit anything from `.idea/` directory if you're not very sure what you doing.

### Delve integration

We're syncing the plugin source with the [Delve](https://github.com/derekparker/delve) debugger.

If you want to customize delve distribution that's used in the plugin you can use `-Dorg.gradle.project.customDlvPath` for setting up the path to your local version of dlv.

### Useful links

For further information please see [the official plugin development page](http://confluence.jetbrains.net/display/IDEADEV/PluginDevelopment).

Also, you can read some [tips and tricks](http://tomaszdziurko.pl/2011/09/developing-plugin-intellij-idea-some-tips-and-links/).

For all development questions and proposals, you can mail to our [Open API and Plugin Development forum](https://devnet.jetbrains.com/community/idea/open\_api\_and\_plugin\_development).

Happy hacking!