# Contributing Code or Documentation to Micronaut

## Finding Issues to Work on

If you are interested in contributing to Micronaut and are looking for issues to work on, take a look at the issues tagged with [help wanted](https://github.com/micronaut-projects/micronaut-core/issues?q=is%3Aopen+is%3Aissue+label%3A%22help+wanted%22).

## JDK Setup

Micronaut currently requires JDK 8

## IDE Setup

Micronaut can be imported into IntelliJ IDEA by opening the `build.gradle` file.

## Docker Setup

Micronaut tests currently require docker to be installed.

## Running Tests

To run the tests use `./gradlew check`.

[Geb](http://gebish.org) functional tests are ignored unless you specify the geb environment via system property.

To run with Chrome `./gradlew -Dgeb.env=chrome check`.

To run with Firefox `./gradlew -Dgeb.env=firefox check`.

## Building Documentation

The documentation sources are located at `src/main/docs/guide`.

To build the documentation run `./gradlew publishGuide` or `./gradlew pG` then open `build/docs/index.html`

To also build the javadocs instead run `./gradlew docs`.

## Building the CLI

- Clone [Micronaut Profiles](https://github.com/micronaut-projects/micronaut-profiles)

- Install micronaut-profiles to Maven Local `micronaut-profiles$ ./gradlew clean publishToMavenLocal`

- `micronaut-core$ ./gradlew cli:fatJar`

- `micronaut-core$ cd cli/build/bin`

- `micronaut-core/cli/build/bin$ ./mn`

## Working on the code base

If you are working with the IntelliJ IDEA development environment, you can import the project using the Intellij Gradle Tooling ( "File / Import Project" and select the "settings.gradle" file).

To get a local development version of Micronaut working, first run the `cliZip` task.

```

./gradlew cliZip

```

Then install SDKman, which is the quickest way to set up a development environment.

Once you have SDKman installed, point SDKman to your local development version of Micronaut.

```

sdk install micronaut dev /path/to/checkout/cli/build

sdk use micronaut dev

```

Now the "mn" command will be using your development version!

The most important command you will have to run before sending your changes is the check command.

./gradlew check

For a successful contribution, all tests should be green!

## Creating a pull request

Once you are satisfied with your changes:

- Commit your changes in your local branch

- Push your changes to your remote branch on GitHub

- Send us a [pull request](https://help.github.com/articles/creating-a-pull-request)

## Checkstyle

We want to keep the code clean, following good practices about organization, javadoc and style as much as possible.

Micronaut uses [Checkstyle](http://checkstyle.sourceforge.net/) to make sure that all the code follows those standards. The configuration file is defined in `config/checkstyle/checkstyle.xml` and to execute the Checkstyle you

need to run:

```

./gradlew <module-name>:checkstyleMain

```

Before start contributing with new code it is recommended to install IntelliJ [CheckStyle-IDEA](https://plugins.jetbrains.com/plugin/1065-checkstyle-idea) plugin and configure it to use Micronaut's checkstyle configuration file.

IntelliJ will mark in red the issues Checkstyle finds. For example:

![](https://github.com/micronaut-projects/micronaut-core/raw/master/src/main/docs/resources/img/checkstyle-issue.png)

In this case, to fix the issues, we need to:

- Add one empty line before `package` in line 16

- Add the Javadoc for the constructor in line 27

- Add an space after `if` in line 34

The plugin also adds a new tab in the bottom to run checkstyle report and see all the errors and warnings. It is recommended

to run the report and fixing all the issues before submitting a pull request.

## Building on Windows 10

The following prerequisites are needed for building and testing on Windows 10:

\* Docker Desktop version 2.0.0.0 win81 build 29211 or higher is installed and running.

\* OpenSSL binaries are installed, for example (https://indy.fulgan.com/SSL/) and on the PATH.