# HowToContribute

## How to Contribute To Hama

### Getting the source code

Download the latest stable Hama source release from http://www.apache.org/dyn/closer.cgi/hama and extract the files. You can get the latest source code using:

```
git clone https://gitbox.apache.org/repos/asf/hama.git
```

#### Eclipse Project File Generation

We are using Maven so you can easily generate project files for our modules.

First you should make sure you have set up your workspace correctly with maven:

```
mvn -Declipse.workspace="/home/user/workspace/" eclipse:configure-workspace
```

**Note:** "/home/user/workspace/" should be the path to your workspace.

You can check if it was successful in Eclipse via:

```
Select Window > Preferences
Select Java > Build Path > Classpath Variables
```

If you see the **M2_REPO** variable, it worked correctly.

Now run the following commands:

```
% mvn install -Phadoop1 -Dhadoop.version=1.2.0
or
% mvn clean install -Phadoop2 -Dhadoop.version=2.0.3-alpha

mvn eclipse:eclipse
```

You can now import the projects into your eclipse workspace via:

```
File -> Import -> Existing Projects into Workspace -> Choose your workspace as the root directory and import
the hama-* projects.
```

### Making Changes

Before you start, send a message to the Hama developer mailing list, or file a bug report in Jira. Describe your proposed changes and check that they fit in with what others are doing and have planned for the project. Be patient, it may take folks a while to understand your requirements.

- Jira usage guidelines

Modify the source code and add some (very) nice features using your favorite IDE.

But take care about the following points

- All public classes and methods should have informative Javadoc comments.
- Code should be formatted according to hama-code-formatter hama-code-formatter fallbacklink
- Contributions should pass existing unit tests.
- New unit tests should be provided to demonstrate bugs and fixes.
- Please resolve all warnings according to "Eclipse Warning Levels" below

### Creating a patch

Check to see what files you have modified with:

```
% git status
```

Add any new files with:

```
% git add any_files_you_created_modified_or_deleted
```

In order to create a patch, type

```
% git diff --cached > /tmp/HAMA-131.patch
```

## Submitting a pull request to GitHub

The Hama apache repository is cloned at the GitHub https://github.com/apache/hama, which is connected to the JIRA issue manager. First, you need to create a GitHub account. Then, you need to fork the Hama master by pushing the fork button on the Hama master. Let say that your GitHub username is xxxxx. You should always bring your local fork up-to-date using the following git commands:

```
# clone your local fork (NOT the Hama master)
git clone https://github.com/xxxxx/hama.git current
cd current
# pull all recent changes from the Hama master
git pull https://github.com/apache/hama.git master
# bring the local fork up-to-date
git push origin master
```

Lets say that you want to create a pull request for the HAMA issue [HAMA-1234] ...issue-title.... You need to create a new branch of your local fork, say named HAMA-1234

```
# create a new branch inside your directory 'current'
git checkout -b HAMA-1234
# ... do some changes to the files ...
# store changes in the branch
git push origin HAMA-1234
# commit changes to the branch
git commit -a -m '[HAMA-1234] ...issue-title...'
Then go to your GitHub HAMA page and do a Pull Request. Use the same title [HAMA-1234] ...issue-title....
```

## Stay involved

Contributors should join the Hama mailing lists. In particular, the commit list (to see changes as they are made), the dev list (to join discussions of changes) and the user list (to help others).

## Coding convention

- Variables should be meaningful so that they are easily understood by other developers. For instance, the declaration of 'String serverName' would be better than 'String str'.
- Indentation is 2 spaces, not 4.
- Argument checks for NullPointerException, IlleagalArgumentException, etc.

### Eclipse Warning Levels

We expect that you turn on following warnings in eclipse:
(can be found in the project preferences or global preferences under Java/Compiler/"Errors/Warnings")

```
From top to bottom:

Non-static access to static member (WARNING)
Indirect access to static member (WARNING)
Method with a constructor name (WARNING)
Parameter assignment (WARNING)
Method can be static (WARNING)

Serializable class without serialID (WARNING)
Assignment has no effect (WARNING)
finally does not complete normally (WARNING)
Using a char array in string concat (WARNING)
hidden catch block (WARNING)
Inexact type match for varags argument (WARNING)
Nullpointer access (WARNING)
Compare identical values (WARNING)
class overrides equals but not hashcode (WARNING)
dead code (WARNING)

Type parameter hides another type (WARNING)
Method does not override package visible method (WARNING)
interface method conflicts with protected object method (WARNING)

Deprecated API (WARNING) // can be neglected by using annotations if not other possible though
Forbidden references (ERROR)
Discouraged references (WARNING)

Value of local variable is not used (WARNING) // please delete it if never read
Unused import (WARNING) // please format and organize imports before making a patch (in eclipse CTRL+SHIFT+F
and CTRL+O)
Unused private member (WARNING) // remove if never used
Unnecessary cast or instanceof operation (WARNING)
Unused break or continue label (WARNING)

Unchecked generic type operation (WARNING)
Usage of raw types (WARNING) // sometimes can not be avoided, can be neglected via annotation
Generic type parameter declared with a final type bound (WARNING)

Missing override annotation (WARNING)
Annotation is used as super interface (WARNING)
Unhandled token in SuppressWarnings (WARNING)
```

## See also

- Apache Contributor Documentation
- Apache Voting Documentation