AutoMergeTool Documentation

Release 0.3.0

Xavier F. Gouchet

Jan 18, 2018

User Documentation

AutoMergeTool is a tool compatible with git to simplify the process of solving conflicts after a git merge, rebase or cherry-pick.

The code is open source, and available on [GitHub](https://github.com/xgouchet/AutoMergeTool).

CHAPTER 1

Introduction

This program allows you to chain different merge tools when resolving conflicts, instead of just using a single (manual) one.

The basic idea is to have (many) simple automatic merging tools that can solve some tedious conflicts, and only launch an interactive tool as a last resort solution.

CHAPTER 2

Origin

After working for a few years in a large team of developers at [Deezer](http://wwww.deezer.com/), merges and rebases came with a recurring sense of fear. The time spent solving conflicts manually seemed like a lot of waste.

After noticing that many conflicts were actually easily resolvable, but tedious, the idea for this tool sparked.

2.1 Installation

2.1.1 Requirements

AutoMergeTool requires Python 3.5. Make sure that your computer uses that version, or higher.

2.1.2 Install using `pip` / `easy_install`

AutoMergeTool is distributed on the PyPi repository, meaning you can install it easily using **pip**:

```
$ pip install automergetool
```

. . . or using **easy_install**:

```
$ easy_install automergetool
```

2.1.3 Configure git

Update your git config, either the global `~/.gitconfig` or the `.git/config` file in a specific repository :

```
[merge]
    tool = amt
    conflictstyle = diff3
[mergetool "amt"]
    cmd = amt -b "$BASE" -l "$LOCAL" -r "$REMOTE" -m "$MERGED"
```

Alternatively, you can just type the following in a shell prompt (ommit the `--global` to only set the configuration for the current repository) :

```
$ git config --global merge.tool amt
$ git config --global merge.conflictstyle diff3
$ git config --global mergetool.amt.cmd 'amt -b "$BASE" -l "$LOCAL" -r "$REMOTE" -m "
↪$MERGED"'
```

2.1.4 Minimal AutoMergeTool configuration

AutoMergeTool requires a minimal configuration. The most basic yet important one is to set the `amt.tools` option in your git config.

```
[amt]
    tools = gen_simplify;gen_additions;meld
```

Alternatively, you can just type the following in a shell prompt (ommit the `--global` to only set the configuration for the current repository) :

```
$ git config --global amt.tools gen_simplify;gen_additions;gen_deletions;meld
```

The above config will launch the `gen_simplify`, then `gen_additions` tool to solve any conflicted file. If neither of those solve all conflicts, then the manual tool (here, `meld`) will be launched.

You can read the configuration page for more details on the configuration, and the Known Merge Tools page for a list of available solvers.

2.1.5 Using AutoMergeTool

Now that AutoMergeTool is configured, whenever you get a merge, rebase or cherry-pick conflict, you can use the following line to automatically solves conflicts.

```
$ git mergetool
```

Note that if you kept another tool as your main `mergetool`, you can run **AutoMergeTool** with:

```
$ git mergetool --tool=amt
```

For more information, read the 'configuration page <configuration>'__, or see the official 'Git Mergetool documentation <https://git-scm.com/docs/git-mergetool>'__

2.2 Configuration

Please note that every configuration described here can be added either to your global `~/.gitconfig` file or to per-project `.git/config` files.

2.2.1 AutoMergeTool

Conflict solvers

The basic AutoMergeTool configuration you need is set the list of solvers which will be applied, in order, on each conflicted file.

When one of the solver solves the last conflict in the file, remaining solvers will be ignored.

Usually it's a good practice to leave your preferred manual solver as the last in the tool chain. eg :

```
[amt]
    tools = gen_simplify;gen_additions;gen_deletions;meld
```

Additional options

You can switch on the `verbose` option to log, on each conflicted file, which solvers are used on which conflict

```
[amt]
    tools = ...
    verbose = true
```

The log of the process will then look like this :

```
Normal merge conflict for 'src/main/Foo.java':
 {local}: modified file
 {remote}: modified file
[AMT] → Trying merge with gen_simplify
[AMT]  gen_simplify didn't solve all conflicts
[AMT] → Trying merge with gen_additions
[AMT]  gen_additions didn't solve all conflicts
[AMT] → Trying merge with gen_deletions
[AMT]  Ignoring tool gen_deletions (unknown tool)
[AMT] → Trying merge with gen_woven
[AMT] ✓ gen_woven merged successfully
```

Keeping reports

Most automatic solvers provided within AutoMergeTool can output reports on solved and unsolved conflicts (see Conflict Reports). By default, AutoMergeTool deletes all report files when all conflicts are solved. You can make AMT keep those reports even when the merge is a success.

```
[amt]
    tools = ...
    keepReports = true
```

2.2.2 Merge Tools

Basic configuration

The basic configuration of merge tools is the same as the standard Git config. Here are the options every mergetool can have:

- `cmd` : a specific command line invocation. The $BASE, $LOCAL, $REMOTE and $MERGED variables will be replace with the path of the corresponding files.
- `path` : for known merge tools, the invocation uses the tool's name. If the tool is not reachable is the Path, you can specify a different path to the executable here. This is usefull when you have different versions of a tool installed on your computer.
- `trustExitCode` : if this option is set to true, then the exit code of the executable will be used as an indication that the merge is fully successfull (or not).

Advanced configuration

In addition to git's standard configuration option, AutoMergeTool can read additional options on each tool to add advanced behavior:

- `extensions` : a semi-colon separated list of file extensions to allow a tool to be used on specific files. Eg: if the `extensions` lists `htm;html`, the tool will only run on HTML files.
- `ignoreExtensions` : a semi-colon separated list of file extensions to prevent a tool to be used on specific files. Eg : if `ignoreExtensions` lists `java;kt`, the tool won't run on java and kotlin files.

Internal solvers

For a detailed list of all internal solvers, and how to configure them, see the Known Merge Tools page.

Custom mergetools

AutoMergeTool has a basic configuration for a few common manual merge tools. But if you want to use an unknown tool, or you want to configure it yourself, you can override the default config using the options described above.

Configuring custom merge tools uses the exact same syntax as the one you're used to on git, and the AutoMergeTool additional parameters are available too.

Here's an example on how you can write those :

```
[mergetool "foo"]
    path = /usr/bin/foo.sh

[mergetool "bar"]
    cmd = bar -o 2 --files $LOCAL $MERGED $REMOTE
    trustExitCode = true
```

Please note that if you provide a `mergetool.<tool>.cmd` value, the `$BASE`, `$LOCAL`, `$REMOTE` and `$MERGED` variables will be replace with the path of the corresponding files.

Also, any unknown option (ie: one that is not listed on this page) that you set in a `mergetool.<tool>` section will be added to the tool invocation. For instance, the following config will append `--auto-merge` to the `meld` invocation command line, and `--config ~/custom_config` when calling `kdiff`

```
[mergetool "meld"]
    auto-merge =

[mergetool "kdiff"]
    config = ~/custom_config
```

For more information on configuring mergetools, see the 'Official ''git-mergetool'' documentation <https://git-scm.com/docs/git-mergetool>'__

2.3 Known Merge Tools

Internal tools refer to automatic conflict solvers bundled with AMT.

External tools refer to 3rd party, usually manual, merge tools. They are known in the sense that you don't have to configure their command line invocation.

2.3.1 Internal (Language agnostic)

Simplify Conflicts (`gen_simplify`)

This tool will handle any conflict for which a large block of code is reported as a single conflict, by splitting it in smaller conflict which can be then each be solved by other tools. It's reccomended to use this as the first tool in the `amt.tools` option.

You can add the following options :

- **mergetool.gen_simplify.report** : sets the type of report (cf Conflict Reports)
- **mergetool.gen_simplify.verbose** : when set to true, logs this solver's process in the console output.

Woven Conflicts (`gen_woven`)

This tool will handle any conflict for which the local and remote modify different lines (eg: local modifies lines 1,2 and 5, remote modifies lines 3 and 4).

You can add the following options :

- **mergetool.gen_woven.report** : sets the type of report (cf Conflict Reports)
- **mergetool.gen_woven.verbose** : when set to true, logs this solver's process in the console output.

Addition Conflicts (`gen_additions`)

This tool will handle any conflict for which the local and remote versions add content in the same place.

You can add the following options :

- **mergetool.gen_additions.order** : sets the preference when adding both sides; can be `remotefirst` (default), `localfirst`, or `ask`
- **mergetool.gen_additions.report** : sets the type of report (cf Reporting)
- **mergetool.gen_additions.verbose** : when set to true, logs this solver's process in the console output.
- **mergetool.gen_additions.whitespace** : when set to true, treats replacement of whitespace and new lines as additions.

Deletions Conflicts (`gen_deletions`)

This tool will handle any conflict for which the local and remote versions both deleted the same content.

You can add the following options :

- **mergetool.gen_deletions.report** : sets the type of report (cf Conflict Reports)
- **mergetool.gen_deletions.verbose** : when set to true, logs this solver's process in the console output.

Single Line Conflicts (`gen_single_line`)

This tool will handle any conflict where a single line was modified both in the local and remote versions, but in different places in the line.

You can add the following options :

- **mergetool.gen_single_line.report** : sets the type of report (cf Conflict Reports)
- **mergetool.gen_single_line.verbose** : when set to true, logs this solver's process in the console output.

2.3.2 Internal (Language specific)

JavaImports (`java_imports`)

This tool will handle any conflict within the imports section of your Java files.

You can add the following options :

- **mergetool.java_imports.order** : specify the way to order imports. Presets include Android Studio : `android`; IntelliJ Idea : `idea`; and Eclipse : `eclipse`

KotlinImports (`kotlin_imports_beta`)

This tool (still in βeta) will handle any conflict within the imports section of your Kotlin files.

You can add the following options :

- **mergetool.java_imports.order** : specify the way to order imports. Presets include Android Studio : `android`; and IntelliJ Idea : `idea`

2.4 Conflict Reports

Most internal tools bundled in AMT have a report feature, that writes a report file next to conflicted files.

2.4.1 Configuration

To enable reporting, you only need to add the following lines in your global `~/.gitconfig` file, or in your project's `.git/config` file.

```
[mergetool "xyz"]
    report = none|solved|unsolved|full
```

The possible values are : - **none** : no report will be written - **solved** : the report file will only log conflicts solved by the `xyz` tool - **unsolved** : the report file will only log conflicts that the `xyz` tool could not solve - **full** : the report file will report both solved and unsolved files

2.4.2 Reporting output

The report file for a file `path/to/foo.ext` will be `path/to/foo.ext.xyz-report`.

Note: AMT automatically deletes report files when a when a conflict is successfully solved. If you want to keep the reports anyway, you can add the following configuration :

```
[amt]
    keepReport = true
```