# Gradle Changelog Automation Plugin Documentation

*Release 0.4.0*

**Zsolt Kovari**

**Mar 20, 2020**

# Contents:

CHAPTER 1

# Overview

## 1.1 Problem to solve

The *Gradle Changelog Automation Plugin* has one purpose: **to automate your CHANGELOG.md generation**.

The plugin addresses the following problems:

- **frequent merge conflicts in your changelog file**
  - even with only 2 developers, merge conflicts are very likely to happen
- **time-consuming manual updates**
  - any time you make a release, you also need to manually update your changelog
- **human errors**
  - due to the manual maintenance, you might make mistakes (e.g. missing entry, wrong version/date, etc.)

## 1.2 Provided solution

Generate your new unreleased changelog entries into separate **YAML** files. One YAML file represents one changelog entry. For example, for a new feature, you would create something like this:

```yaml
title: My new feature
reference: GH-1
author: John Doe
type: added
```

Then, during release time, use this plugin to automatically process your unreleased entries and combine them into your `CHANGELOG.md` file:

```
## [1.0.0] - 2019-07-21
### Added
- GH-1 My new feature (John Doe)
```

## 1.3 Followed conventions

The plugin generates a `CHANGELOG.md` file in the root of the project. The content of the generated changelog is based on Keep a Changelog.

The unreleased changelog entries are expected to be under `changelogs/unreleased` directory in the root of the project.

The generator script that generates YAML files is expected under `scripts/changelog.sh`.

## 1.4 Original source of idea

This plugin was inspired by GitLab:

- https://about.gitlab.com/2018/07/03/solving-gitlabs-changelog-conflict-crisis/

CHAPTER 2

# User Guide

## 2.1 How to apply the plugin

Groovy

Using the plugins DSL:

```
plugins {
    id 'org.zkovari.changelog' version '0.4.0'
}
```

Using legacy plugin application:

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'org.zkovari.changelog:changelog-automation-gradle-plugin:0.4.0'
    }
}

apply plugin: 'org.zkovari.changelog'
```

Kotlin

Using the plugins DSL:

```
plugins {
    id("org.zkovari.changelog") version "0.4.0"
}
```

Using legacy plugin application:

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath("org.zkovari.changelog:changelog-automation-gradle-plugin:0.4.0")
    }
}

apply(plugin = "org.zkovari.changelog")
```

## 2.2 How to generate unreleased changelog entries (YAML files)

To create new unreleased changelog entries, the easiest way if you use the plugin's built-in generator script. It is already on the classpath, you just have fetch it with the `fetchChangelogScript` task:

```
gradle fetchChangelogScript
# result: scripts/changelog.sh

# also add permission
chmod +x scripts/changelog.sh
```

As a result, the script is present under `scripts/changelog.sh`. To generate a new entry, run `changelog.sh`:

```
./scripts/changelog.sh --type added "My new feature"
```

As a result, a new changelog entry is generated under `changelogs/unreleased`.

For type (`-t|--type`) the following values are accepted: `added`, `changed`, `deprecated`, `fixed`, `removed`, `security`. Shortened values can be also specified, for example, `a` corresponds to `added`, `secu` to `security`, etc.

See `changelog.sh --help` for more information.

### 2.2.1 Optionally specify reference and author

Optionally a reference (`-r|--reference`) or the author (`-u|--git-username`) can be also specified in the unreleased entry. The reference could typically refer an issue or a pull/merge-request number. For author, the Git username is used (from Git config). E.g. running:

```
./scripts/changelog.sh --type fixed -u -r "13" "Fix bug"
```

... would create:

```
title: Fix bug
reference: 13
author: zkovari
type: fixed
```

## 2.3 How to process unreleased entries into CHANGELOG.md

If you already have unreleased YAML entries under `changelogs/unreleased`, you can combine them into your `CHANGELOG.md`. The generated changelog is based on Keep a Changelog.

To process the unreleased entries, run the task `processChangelogEntries`:

```
gradle processChangelogEntries
```

Result is `CHANGELOG.md`. The unreleased entries are also automatically removed from `changelogs/unreleased`.

**`CHANGELOG.md`**

```
# Changelog
All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](https://keepachangelog.com/en/1.0.0/),
and this project adheres to [Semantic Versioning](https://semver.org/spec/v2.0.0.
↪html).

## [1.0.0] - 2019-07-21
### Added
- My new feature
```

For version, always the project's version, while for date, ISO standard format it used: `YYYY-MM-DD`.

### 2.3.1 Continuous processing

New release entries can be continuously generated. In that case, the previous `CHANGELOG.md` will be updated with a new released changelog.

As an example, see this project's changelog.

CHAPTER 3

# Supported Java and Gradle Versions

**Java** Java 8+ is required.

**Gradle** We guarantee backward-compatibility with the latest 3 major Gradle versions, and we provide best effort compatibility with prior versions.

We daily integrate with different Gradle and Java versions. Currently, the following combinations are tested:

| Gradle | Java | Compatibility |
|---|---|---|
| latest version | 8/11/12 | Guaranteed as soon as possible |
| 6.0 | 8/11 | Guaranteed |
| 5.0 | 8/11 | Guaranteed |
| 4.10.3 | 8/11 | Guaranteed |
| 4.0 | 8 | Guaranteed |
| 3.5 | 8 | Best effort, might break in the future |
| 3.0 | 8 | Best effort, might break in the future |
| 2.14.1 | 8 | Best effort, might break in the future |