

Using Git with Sling

This page documents decisions we made while finding out how to best work with Git as well and various quality-of-life tips related to Git.

Cloning the Sling git repositories

This is documented in the [Sling Aggregator Github module](#).

Gitiquette

Force pushing to the master branch, or to other branches that are shared with others, is discouraged. It makes collaboration harder and potentially impacts other tools as well - Jenkins, SonarCloud, etc. If you need time experimenting and fiddling with history, it's better to start working with an individual repository and then bring it to Sling, once it's stable.

Merging pull requests from contributors

Maintaining a linear history in git is preferred, especially for small contributions. Each commit should be buildable and correct by itself, as much as possible. This is beneficial when reviewing history for changes and also when running `git bisect`.

In practice, we have found that:

- for contributions which require no additional fixes we should use the 'Rebase and merge' workflow
- for contributions which require fixes
 - in case of a single logical commit we should use the 'Squash and merge' workflow to create a single commit. This creates a fast-forward merge, so no merge commit.
 - in case of multiple separate commits we should ask the reviewer to amend the original commits with the fixes and force-push the branch that is the source of the PR
- using the 'merge' button for PRs should be the exception

It is also important to maintain the original author of the pull request, giving them credit for their contribution. This is also useful when compiling various contribution metrics, e.g. in Kibble.

Accepting new significant contributions




When accepting an external contribution we will import it into the `sling-whiteboard` repository. Before making a new release, we will move it to its own repository with the release manager also proposing a proper artifact id and therefore repository name.

It is recommended to provide ownership information when accepting new contributions by maintaining the original *Author* git header.

See also [dev@sling - Where do we put new git modules?](#)

Code donations

In case of code developed externally and not initially intended to be hosted at the ASF IP clearance is required. The general rules are outlined at <http://incubator.apache.org/ip-clearance/>. For a hands-on example, the Sling Journal-based Content Distribution is a good example:

- Incubator IP clearance page: <http://incubator.apache.org/ip-clearance/sling-distribution-journal.html>
- Top-level Jira:  **SLING-8326** - Donation proposal of Journal based Sling Content Distribution CLOSED
 - IP clearance sub-task:  **SLING-8345** - IP clearance of Journal based Sling Content Distribution donation RESOLVED
 - Source code import sub-task:  **SLING-8346** - Import Journal based Sling Content Distribution source code RESOLVED

Creating new repositories

Git repository creation

Repository creation is open only to PMC members. If you are not part of the PMC please ask on dev@sling.apache.org and someone will do this for you.

When creating new repositories we follow the convention of deriving the repository name from the Maven artifactId. The full logic can be found in the [migrate-to-git.sh script](#), but the gist is:

1. Append `sling-` to the artifact id if not there (required by ASF infra conventions)
2. Replace all dots with dashes (ASF infra does not allow dots in repository names)

It is recommended to announce the intention on the mailing list, without calling a formal vote, since renaming repositories requires ASF infra manual intervention.

Creating a new repository is done using the web UI at <https://gitbox.apache.org/>. Please fill in the fields in the following manner:

- **Name:** artifact Id , dots replaced with slashes. Should not contain the `sling-` prefix as the form will auto-insert it in the 'Generated Name' field
- **Repository Description:** name from the pom.xml, otherwise short summary of the form *Apache Sling Foo*
- **Commit Notification List:** commits@sling.apache.org
- **GitHub Notification List:** commits@sling.apache.org

See a recent example below

PMC:	sling ▼
Repository name:	org-apache-sling-distribution-joi
Generated name:	sling-org-apache-sling-distributi
Repository description:	Apache Sling Journal based Content Distribution - Core
Commit notification list:	commits@sling.apache.org
GitHub notification list:	commits@sling.apache.org

Submit request



Default branch name

Sling uses the default branch name `master` . To keep things simple for our tooling (repo tool, Jenkins) please create the repository locally and use `master` as a branch name when pushing the initial changes to the repository. See [Removal of problematic language](#) for more context.

```
$ mkdir sling-new-repo
$ cd sling-new-repo
$ git init -b master
```

See also the discussion from



[INFRA-22359](#) - Jira project doesn't exist or you don't have permission to view it.

Permission propagation

The scripts will be created in gitbox and github at the same time, but permissions will be granted to the GitHub repository sometime later, due to the way ASF infra tooling works. If permissions don't appear in two hours, [raise an INFRA ticket](#). In the meantime you can use the gitbox backend to push your changes.

Importing existing code from the whiteboard

If you have code that exists in the whiteboard and you want to move it to a separate repository it is recommended to keep the history. The steps to achieve that are below, given that `${FOLDER}` is the folder from the Sling Whiteboard that you want to move to a new repository, and that `${NEW_REPO}` is the name of the new repository. The new repository should already be created.

1. Clone the whiteboard repository

```
$ git clone https://github.com/apache/sling-whiteboard.git
$ cd sling-whiteboard
```

2. Reduce the repository to only the subfolder you're interested in

```
$ git filter-branch --prune-empty --subdirectory-filter ${FOLDER}
```

3. Change the origin remote URL to the desired one

```
$ git remote set-url origin https://github.com/apache/sling-${NEW_REPO}.git
```

4. Review your changes and push them

```
$ ls -l  
$ git log  
$ git push -u origin master
```

Reference: <https://help.github.com/en/articles/splitting-a-subfolder-out-into-a-new-repository>

Maven SCM settings

The SCM settings from the pom.xml file are read by various tools, but most often used by the `maven-scm-plugin` when releasing. We set the connection and developerScm settings to reference the ASF-maintained gitbox service, but use GitHub for the URL since it offers a better browsing experience and receives special treatment from some tools, such as renovate.

pom.xml scm settings example

```
<scm>  
  <connection>scm:git:https://gitbox.apache.org/repos/asf/sling-org-apache-sling-${MODULE}.git</connection>  
  <developerConnection>scm:git:https://gitbox.apache.org/repos/asf/sling-org-apache-sling-${MODULE}.git<  
/developerConnection>  
  <url>https://github.com/apache/sling-org-apache-sling-${MODULE}.git</url>  
</scm>
```

Update the repo manifest

Regenerate the default.xml manifest from the [sling-aggregator repo](#) using `groovy collect-sling-repos.groovy > default.xml`.

Update local Sling repo checkout

Execute the following command in the root of the Sling repo checkout

```
$ repo sync -j 16  
$ cd ${NEW_REPO}  
$ git checkout master
```

Boilerplate files

Most of the files can be copied from any repository. We'll use `org-apache-sling-api` as an example.

```
$ cd org-apache-sling-api  
$ cp CODE_OF_CONDUCT.md CONTRIBUTING.md Jenkinsfile LICENSE .gitignore ../${NEW_REPO}
```

Generate a default `.asf.yaml` file and update it, if applicable:

```
$ cd aggregator  
$ groovy ./scripts/update-asf-yaml.groovy ../org-apache-sling-commons-metrics-prometheus/
```

See [Git - .asf.yaml features](#) for more details:

- tags: include initially `sling` and `java` (or other programming language, as applicable)
- homepage: module documentation on Sling website, or the Sling homepage if none exists

- description: the name from the `pom.xml`

Badges

The `sling-aggregator/add-badges.sh` script automates the creation of badges for the README. You will need a valid GitHub token for it. To add the badges for just a single repository, run.

```
$ cd aggregator
$ groovy ./generate-project-badges.groovy ../ ${NEW_REPO}
```

Enroll in Kibble

See [Repository analysis with Kibble](#) .

Onboard to SonarCloud

See the onboarding section in [SonarCloud analysis](#) .

Customize GitHub settings

By placing an [.asf.yaml](#) file one can customize certain GitHub settings like label, features and merge options.

Creating links to Git commits

To quickly create links to Git commits in JIRA a script such as the one below can be used:

jira_link_for_commit.sh

```
#!/bin/sh -e
if [ $# -eq 0 ]; then
    commit=HEAD
else
    commit=$1
fi
hash=$(git rev-parse --short ${commit})
base=$(git remote get-url origin)
url=${base%.git}/commit/${hash}
echo "[commit ${hash}] | ${url}]"
```