A Plan for Maintaining Spack Recipes

Version 1

2025-03-25

1 Introduction

This document proposes a plan for CSAID maintenance of Spack recipes.

2 Preconditions

2.1 Fork of the Spack recipe repository

CSAID should maintain a fork of the main Spack repository that contains recipes. [1]

The reason for having the fork is to allow us to modify recipes which we need to modify on our own time scale. The Spack team sometimes delays the merging of a pull request (PR) into the Spack recipe repository. By maintaining our own fork, we can make acceptable locally-originated changes available to our users more quickly than otherwise would be the case.

Maintenance of the fork is not free. The main Spack recipe repository is updated frequently[2], and so we will need to automate the updating of the fork. It is probably sufficient to update the fork daily, but it is always possible to manually trigger updates when needed.

We anticipate that few recipes will need modification in our fork, and thus automatic updating of the fork will rarely lead to merge conflicts. When conflicts do occur, whoever has modified the recipe should be responsible for reconciling the recipe.

In common *git* parlance, the original repository from which a fork is created is called an *upstream* repository. The act of pushing commits from the fork to the upstream repository is called *upstreaming* the commits. We use this terminology in the rules below.

See rules 3–5 below for conditions under which CSAID's recipe repository fork will differ from Spack's upstream recipe repository.

2.2 One recipe repository per CSAID project

The Spack recipe for a given package must not live in the same repository as the source code for that package to avoid excessive and unnecessary changes in the recipe. All known efforts in CSAID currently using Spack already follow this guideline.

CSAID should maintain one repository for each project[3], to contain the Spack recipes for the project. Keeping all the recipes for a given project in a single recipe repository provides some cohesion to the package, and will allow each project to have its own accounting of GitHub actions "minutes" for continuous integration usage.

2.3 No cyclic dependencies between recipe repositories

We should take care to ensure that the dependency graph of recipe repositories (as implied by the dependencies between the recipes) remains a directed acyclic graph. If adding a new recipe to a repository would create a dependency cycle between repositories, the preferred approach to keeping things acyclic is to add the recipe instead upstream, or to move recipes between existing recipe repositories rather than creating new recipe repositories.

Spack itself does not require this, but doing this will make it easier to ensure we do not introduce any cycles into the dependency graph of the packages themselves.

[1] Before the release of Spack v1.0, a single repository contains both the source code for Spack and a set of Spack recipes for many products. As part of the v1.0 release, all recipes will be removed to a separate repository.

[2] For example, the GitHub statistics for the spack repository reports 364 merged pull requests over the period January 18, 2025 through February 18, 2025. Most of these are changes to recipes.

[3] For our purposes, a *project* is a set of Spack-managed packages (i.e. software repositories) that are maintained in concert, and which are typically used together. Some examples are: the *art* project, *LArSoft*, and *FIFE*. A repository maintained by a small team which has no other closely associated packages may be a project by itself, e.g. *MARLEY*.

3 Rules for maintaining Spack recipes

1. If a package is developed under the control of a specific project, its recipe shall be maintained in a recipe repository appropriate to the project, and distinct from that of the package source.
2. If a recipe for a required package exists in Spack's upstream repository it should be used from our fork of the Spack recipe repository.
3. If alterations are required to an upstream recipe, those alterations shall be upstreamed. If the maintainers of the Spack recipe repository do not accept a pull request on a suitable timescale, an equivalent PR may also be submitted to the CSAID fork with the understanding that the submitter is responsible for reconciliation as soon as reasonably practicable.
4. If a recipe for a package not maintained at Fermilab does not exist in Spack's upstream repository, then the new recipe shall be upstreamed if its applicability may reasonably be expected to be outside one experiment/project. Packages maintained at Fermilab may have their recipes upstreamed if there is an expectation they might be used elsewhere.
5. If a new recipe would be upstreamed but for overriding time constraints, then it may be submitted in parallel to CSAID's fork with the understanding that the submitter is responsible for reconciliation as soon as reasonably practicable, and that upstream will be the primary venue for maintenance going forward.