

Continuous integration and testing for Proof General

Contents

1	Introduction	1
2	Generic strategy	2
3	Container build strategy	2
4	Testing strategy	4
4.1	Compilation and indentation	4
4.2	Up-to-date documentation strings in the manuals	4
4.3	Proof General interaction tests with Coq	4
5	Keeping the GitHub action up-to-date	5
5.1	Release table	6

1 Introduction

This document describes the general strategy for testing Proof General with GitHub actions and for building the necessary containers.

The file `.github/workflows/test.yml` defines 5 jobs.

build compile Proof General sources to bytecode and build documentation

check-doc-magic: check that the documentation strings from the source code that are magically included in the manuals are correct

test run the tests in `ci/coq-tests.el`

compile-tests run the auto-compilation tests in `ci/compile-tests`

simple-tests run the tests in `ci/simple-tests`

test-indent run Coq source code indentation tests

The jobs `build`, `check-doc-magic` and `test-indent` run on different Emacs versions but they do not need Coq. They therefore use the GitHub hosted `ubuntu-latest` runner together with the `purcell/setup-emacs` action to install Emacs.

The jobs `test`, `compile-tests` and `simple-tests` need Coq and Emacs in different version pairs. They therefore use the `proofgeneral/coq-emacs` containers build specifically for various Coq/Emacs version pairs for this purpose. This is achieved by installing Nix on specific containers from `coqorg/coq`, see

`proofgeneral/coq-nix`, and then installing Emacs via Nix. There are GitLab and GitHub projects for building these containers in GitLab pipelines. However, currently this does not work and the containers are therefore build manually.

This document ignores Coq point releases (e.g., 8.13.2). For one reason we trust the Coq development team to not introduce changes in point releases that Proof General should care about. Another reason is that the `coqorg/coq` docker images are only available for the latest point release of any minor version. Therefore, in this document, a Coq version number `<major>.<minor>` always stands for the latest point release of that minor release. E.g., 8.17 stands for 8.17.1 released in 2023/06.

2 Generic strategy

Proof General distinguishes between active and passive support for certain Coq/Emacs versions pairs. Active support means that tests for these Coq/Emacs versions pairs run on each pull request and the pull request should not be merged if one of these tests fails unexpectedly. Passive support means that the developers refrain from knowingly breaking the compatibility with passively supported version pairs. (In the future we plan to also run tests on passively supported version pairs.)

The actively supported Coq/Emacs version pairs are all Coq/Emacs version combinations starting from the oldest Coq and the oldest Emacs version from all Debian and Ubuntu releases that are still enjoying standard support. For Debian these are the releases whose end of live date is in the future. For Ubuntu these are the releases whose end of standard support date is in the future.

Currently, the first actively supported versions are

Coq	8.11
Emacs	26.3

The set of passively supported Coq/Emacs version pairs is work in progress.

In May 2023, shortly before Ubuntu 18.04 Bionic Beaver reached its end of standard support, all Coq versions since 8.6 and all Emacs versions since 25.2 were actively supported, resulting in 12 major Coq versions, 9 Emacs versions and 108 version pairs.

To keep test runtime and space for containers reasonable, we build containers only for a subset of the actively supported version pairs and only a subset of these containers are used by the GitHub action jobs.

3 Container build strategy

For an actively supported Coq/Emacs version pair (cv, ev) , a container containing that Coq and Emacs version is built, in case one of the following points is true for cv and ev .

1. The Emacs version *ev* is present in an Debian or Ubuntu release still enjoying standard support and the Coq version *cv* or an earlier Coq version is present in the same release.

This point is motivated by the ability to reproduce problems in a supported Debian or Ubuntu installation in which Coq has been updated via opam.

2. The Coq version *cv* has been released within the last two years.

This point is motivated by the ability to reproduce Problems for recent Coq versions.

3. The pair (*cv*, *ev*) is an historic pair, in the sense that both versions were for several month the newest versions in the past. E.g., in the second half of 2020 Coq 8.12 and Emacs 27.1 were respectively the newest versions for about 6 month. Therefore the pair (8.12, 27.1) is considered an historic pair.

This point is motivated by the ability to reproduce Problems for versions that may have been used by many people for several month in the past.

4. The Coq version *cv* is the newest Coq version or the Emacs version *ev* is the newest Emacs version.

This point is motivated by the compatibility of newest versions.

5. The Coq version *cv* is a release candidate.

This point is motivated by the compatibility of release candidates.

Eventually we will also spell out the container build strategy for the passively supported versions. For now, in addition to the rules above, we build containers for the historic pairs of the last 6 years as passively supported versions.

This results in 43 containers.

	25.3	26.1	26.2	26.3	27.1	27.2	28.1	28.2	29.1
8.7	H								
8.8		H							
8.9			H						
8.10				H					
8.11				SUP					N
8.12				SUP	SUP				N
8.13				SUP	SUP	H			N
8.14				X	X	X	X	X	X
8.15				X	X	X	X	X	X
8.16				X	X	X	X	X	X
8.17				X	X	X	X	X	X
8.18				X	X	X	X	X	X

In the table above,

SUP denotes a supported pair according to point 1. above.

X denotes a Coq release within the last 2 years according to point 2. above.

H denotes an historic pair according to point 3.
N denotes newest Coq or Emacs versions according to point 4.
RC denotes an release candidate according to point 5.

4 Testing strategy

Currently only actively supported versions are tested via GitHub actions.

4.1 Compilation and indentation

The tests defined for the build and test-indent jobs run for all actively supported Emacs versions.

4.2 Up-to-date documentation strings in the manuals

The error scenario here is that a documentation string that is magically included in the manuals has been updated without updating the manuals via `make magic`. It is therefore sufficient to run the test whether the manuals are up-to-date with only the latest two Emacs versions.

4.3 Proof General interaction tests with Coq

The tests for the GitHub actions test, compile-tests and simple-tests can obviously only run for those version pairs for which containers have been build, see [Container build strategy](#). To limit the required resources, the tests only run on a subset of the available containers.

The jobs test, compile-tests and simple-tests run for an actively supported Coq/Emacs version pair (cv, ev) , in case one of the following points is true for cv and ev .

1. The version pair (cv, ev) is present in an Debian or Ubuntu release still enjoying standard support.

This point is for Proof General users using one of these currently supported releases.

2. The Emacs version ev is present in an Debian or Ubuntu release still enjoying standard support and the Coq version cv has been released in the last 18 month and is either equal or greater than the Coq version of this Debian or Ubuntu release.

This point is for Proof General users that use one of the supported Debian or Ubuntu versions and now and then upgrade their Coq version via opam.

For example, in October 2023 the 18 month limit includes Coq 8.15 (last minor version released in 2022/05) but excludes 8.14 (last minor version released in 2021/12). Therefore we do test the pair (8.15, 27.1) but we don't test (8.14, 27.1). We neither test (8.15, 27.2), because there is no Debian or Ubuntu release containing 27.2. Further, we don't test (8.15, 28.2), because Ubuntu 23 and Debian 12, which both contain Emacs 28.2, contain Coq 8.16.

3. The pair (cv, ev) is an historic pair in the same sense as above.

This point is motivated by the compatibility with version pairs that may have enjoyed some wider use.

4. The Coq version cv is the newest Coq version or the Emacs version ev is the newest Emacs version.

This point is motivated by the compatibility of newest versions.

5. The Coq version cv is a release candidate.

This point is motivated by the compatibility of release candidates.

Running Proof General interaction tests with Coq for passively supported versions is work in progress.

This results in 26 version pairs for the Proof General interaction tests with Coq.

	25.3	26.1	26.2	26.3	27.1	27.2	28.1	28.2	29.1
8.7									
8.8									
8.9									
8.10									
8.11				SUP					N
8.12					SUP				N
8.13						H			N
8.14						H			N
8.15				X	X		H		N
8.16				X	X			SUP	N
8.17				X	X			X	SUP
8.18				X	X	N	N	X	X

See [Container build strategy](#) for an explanation of the symbols in the table.

5 Keeping the GitHub action up-to-date

Obviously, after each Coq or Emacs release and additionally every few month the rules in the preceding sections for building containers and for testing must be re-evaluated and the workflow file `.github/workflows/test.yml` must be updated.

Large portions of this process have been automated. Coq, Emacs, Debian and Ubuntu releases must be manually added into an Org mode table in file `coq-emacs-releases.org`. This table is read by the OCaml program `cipg` that performs the following tasks.

- It evaluates the rules spelled out above and generates the tables in the Sections [Generic strategy](#), [Container build strategy](#), and [Proof General interaction tests with Coq](#) in this document.
- It determines missing docker images and generates command lines for building them.

- It determines superfluous docker images and deletes them.
- It generates the lines that are needed to update `.github/workflows/test.yml`.

5.1 Release table

The release table is the Org mode table in file `coq-emacs-releases.org`. It contains all the data that is needed to evaluate the rules for building containers and for testing in the preceding sections. In particular, it contains

- Coq, Emacs, Debian and Ubuntu releases with their release date.
- The end of standard support for Debian and Ubuntu releases.
- The Coq and Emacs versions contained in these Debian and Ubuntu releases.
- The historic pairs.

The remainder of this section explains how to maintain this table such that `cipg` can process it.

1. Whenever a new version of Coq, Emacs, Debian or Ubuntu is released, a line needs to be added manually to the release table.
2. The date must be present in every line in the form YYYY/MM.
3. There are 3 kinds of lines in the table.

Coq releases are specified with a date and a Coq version.

Emacs releases are specified with a date and an Emacs version.

Debian or Ubuntu releases are specified with a date, a distribution name and an EOL date.

4. The Coq and Emacs versions of an Debian or Ubuntu release may be omitted. If they are not present, they are taken from the last preceding line containing the respective version (the table is processed from bottom to top). E.g., the line specifying the release of Ubuntu 20.4 (on 2020/04) inherits Coq version 8.11.0 from the preceding line and Emacs version 26.3 from 5 lines below.
5. The lines do not need to be sorted by date. I find it clearer to insert Debian and Ubuntu releases at a place where they can inherit the Coq and maybe the Emacs version from a previous line, such that these version numbers can be omitted. E.g., the line for Debian 11 Bullseye released in 2021/08 appears before the Coq release 8.12.1 in 2020/11, because Bullseye was released with 8.12.0.

Sometimes it is impossible to move a Debian or Ubuntu release line to a place where it can inherit both, the Coq and Emacs version, without scrambling the table too much. E.g., Ubuntu 22 was released with Coq 8.15.0 and Emacs 27.1. In this case I prefer to keep the table sorted by Coq releases, to specify the Emacs version for Ubuntu 22, and move the line to a place where it can inherit the Coq version. The Emacs column therefore contains some discontinuities (together with the date column).

6. Historical pairs must be manually marked in column `historic`. They are mainly used to manually select some additional version pairs for testing

and to have some sort of diagonal in the version matrix for Proof General interaction tests.

7. The EOL date of Debian or Ubuntu releases must also be in the form YYYY/MM. Trailing non-digit characters are ignored. I use ? to indicate EOL dates that have not yet been fixed.
8. In Coq version numbers the patch level is ignored. Versions of release candidates must be of the form 8.10rc or 8.10-rc. `cipg` is not able to recognize outdated release candidates. The release candidate must therefore be deleted when the release happens.