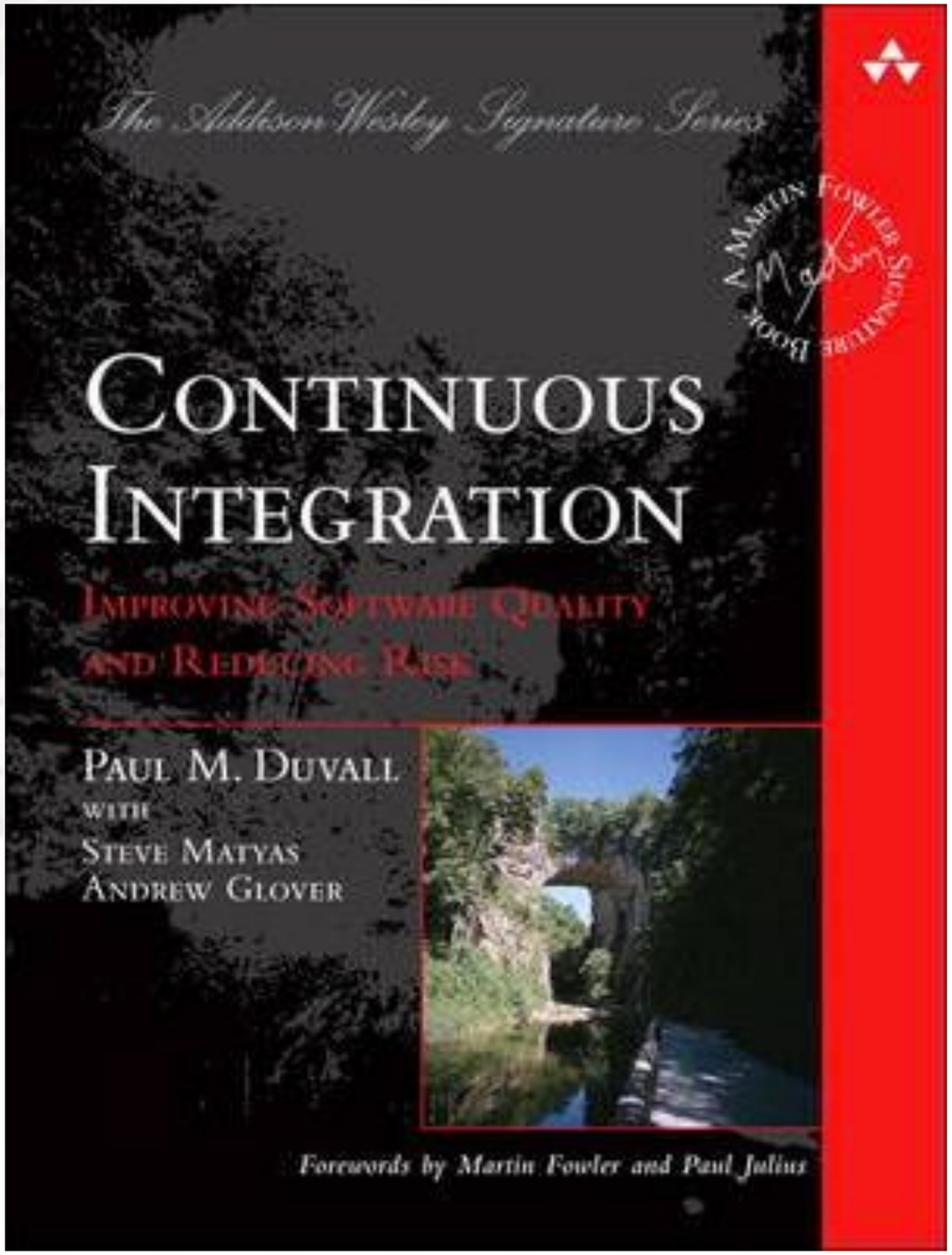
Continuous Deployment

Continuous Integration



Continuous Deployment

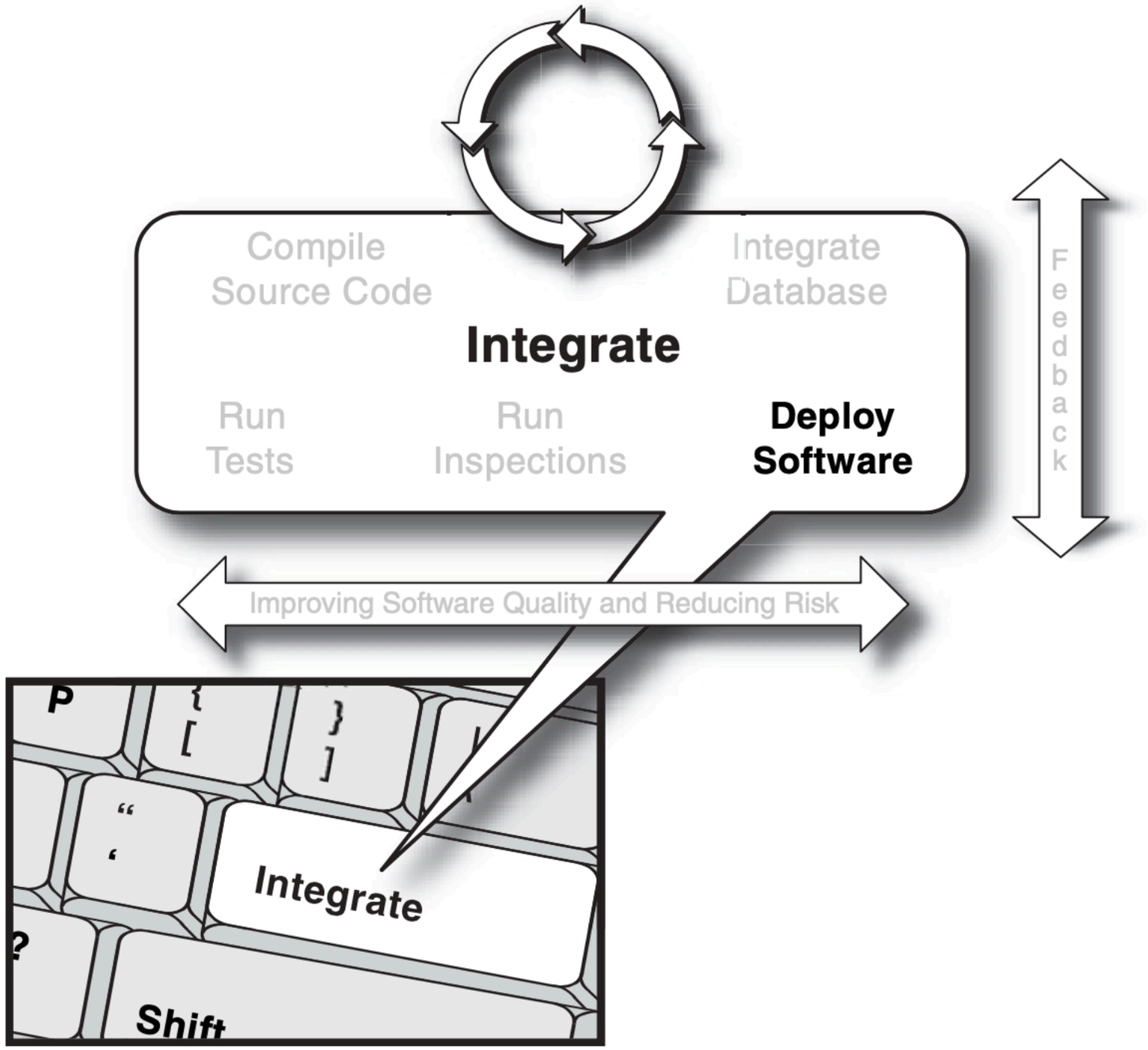


FIGURE 8-1 Integrate button—deploy software

Release Working Software Any Time, Any Place

Deploying working software principally embodies six high-level steps.

1. Label a repository's assets.

2. Produce a clean environment, free of assumptions.

3. Generate and label a build directly from the repository and install it on the target machine.

4. Successfully run tests at all levels in a clone of the production environment.

5. Create build feedback reports.

6. If necessary, you can roll back the release by using labels in your version control repository.

Label a Repository's Assets

"Creating a repository label facilitates the identification and tracking of assets, as it clearly delineates a group of files as belonging together. "

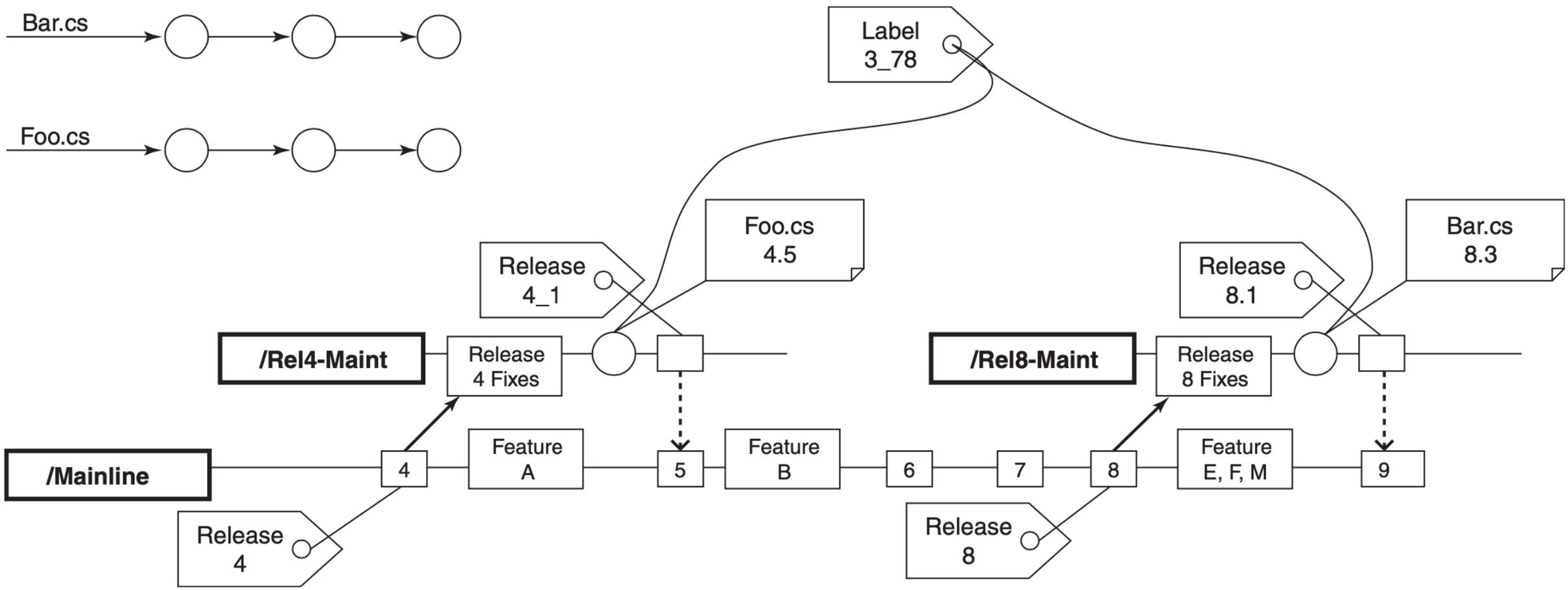
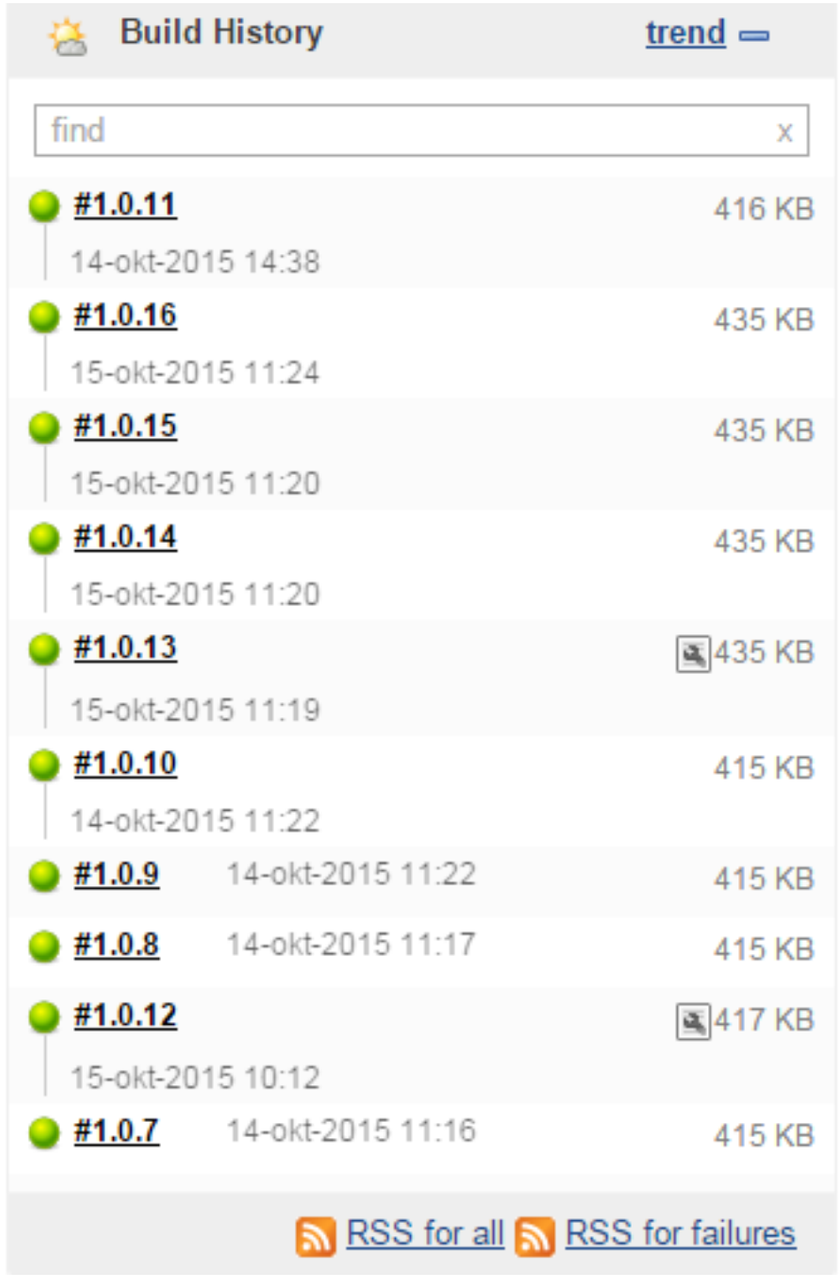


FIGURE 8-2 Foo.cs and Bar.cs in the same repository

Produce a Clean Environment

Label Each Build



Run All Tests

Create Build Feedback Reports

#18

Bygge #18 (2015-nov-20 22:33:57)

Git Changelog changelog

Changelog of Git Changelog.

pull-request-notifier-for-bitbucket-2.12

GitHub #82

c19f72f04d33d9e Tomas Bjerre *2015-11-09 16:36:20*

Fixing PULL_REQUEST_URL-bug correctly with getSlug #82

No issue

Expand

0cd2c14c0b1f1e3 Tomas Bjerre *2015-11-09 16:28:05*

Renaming application variable to lowercase

pull-request-notifier-for-bitbucket-2.11

GitHub #82

1e2d237a8c565b9 Tomas Bjerre *2015-11-06 19:27:17*

Replacing spaces with dashes in PULL_REQUEST_URL #82 * Was evaluating to wrong URL if repo name included spaces.

pull-request-notifier-for-bitbucket-2.10

GitHub #78

eef94cd53904b7e Tomas Bjerre *2015-10-16 15:48:32*

Processing events on Bitbucket Server's event threads #78

pull-request-notifier-for-bitbucket-2.9

GitHub #76

Possess Capability to Roll Back Release

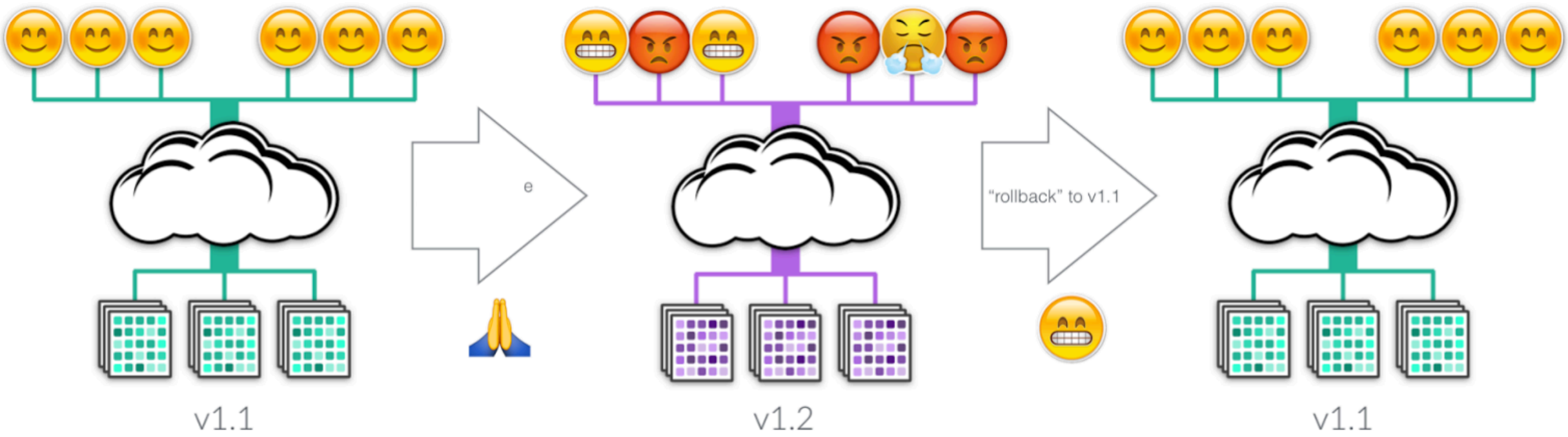


TABLE 8-1 CI Practices Discussed in This Chapter

Practice	*Description*
Release working software any time, any place	By running a fully automated build including compilation, all tests, inspections, packaging, and deployment, you have the capability to release working software at any time and in any known environment.
Label a repository's assets	Label the files for your project in your version control repository. Typically, this is performed at the end of a project milestone.
Produce a clean environment	Remove all files, configuration changes, servers, and anything else from your integration build machine and ensure you can rebuild back to a state where your integration build is successful. The more scripted this process is, the better.
Label each build	Label the binary artifacts of a build distribution in your version control repository.
Run all tests	Run all tests against the software. This includes unit, component, system, functional, and perhaps even performance, load, and other types of tests that ensure the software is ready to be delivered (to the next stage or even to production).
Create build feedback reports	List the changes that were made in the most recent build. This can be useful for other teams in the delivery process, such as QA.
Possess capability to roll back release	Something can always go wrong, so use your build labels to roll back any changes that shouldn't have been committed to the version control repository.