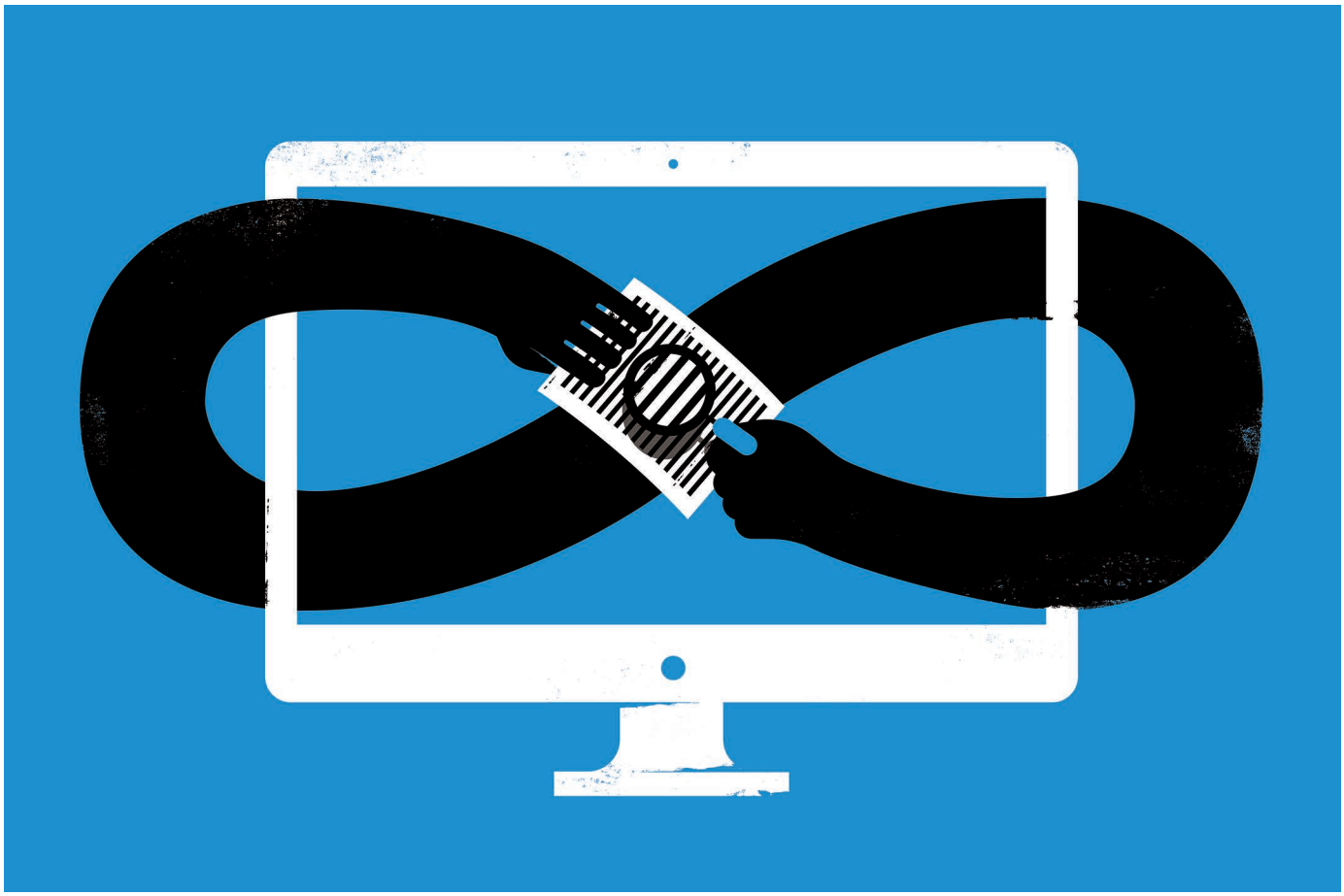
TOOLBOX

COLLABORATIVE SOFTWARE DEVELOPMENT MADE EASY

Save time and protect critical code with 'continuous integration' services.

ILLUSTRATIONS BY THE PROJECT TWINS



BY ANDREW SILVER

Sebastian Neubert, a particle physicist at Heidelberg University in Germany, leads a group studying subatomic particles called pentaquarks. The six team members all have access to the software code used to run their multi-step analyses, and the programmers update it daily with new features and bug fixes. With each code change, however, they run the risk of introducing inadvertent errors that foul the underlying algorithms.

To prevent that, the team checks and rechecks the analyses, and uses error-checking algorithms, functions they can call whenever a change is proposed, to ensure that their software works as intended. One test, for example, verifies that a noise-cancelling algorithm gives the correct output when it is run on practice data.

In 2015, in an effort to save time and resources, the team took inspiration from the technology industry, automating their testing using a process called 'continuous integration'.

In continuous integration, changes to software code automatically trigger repetitive tasks, such as error-checking. Fundamentally, the process simplifies a task that diligent coders already perform. Programmers usually write lists of tests that they will run periodically to ensure that their code still works, just as Neubert's team do. But a busy team might forget or lack the time to run them, allowing errors to creep in. Continuous integration automates that process so those checks run whenever a change is proposed, saving team members the time they would spend hunting down an error. A team running genomic analyses could spend more time at the bench, while a group developing climate-prediction software could better refine its models. That said, the resulting peace of mind is only as good as the tests themselves: a poorly designed test can still allow mistakes to pass undetected.

The process is common in the commercial and open-source sectors. A study presented at the 2016 IEEE/ACM International Conference on Automated Software Engineering ▶

Toolbox in the blogosphere

Nature's technology editor, Jeffrey Perkel, started blogging about workplace technology in science on the Naturejobs website in 2016. Here are some highlights.

From stadiums to genomes
Most bioinformaticians are either biologists skilled in programming or programmers with an interest in biology. Mike Goodstadt, the programmer behind the 3D genome-visualization tool TADkit, took a different approach. In the early-to-mid 1990s, Goodstadt was a student at the University of Bath, UK. His course of study? Architecture, with an emphasis on 3D modelling. After graduation, he helped to design and build a 61,500-seat stadium. But a faltering economy and newly acquired programming skills helped to steer him towards biology.
go.nature.com/2yaweu8

Lorena Barba, reproducibility champion
Lorena Barba, a mechanical and aerospace engineer at George Washington University in Washington DC, has long championed research reproducibility. "I've always believed that the open-source model is ideal for science, as it exposes the complete sequence of steps that produces a given result," she says. In January, she travelled to Chile to run a week-long course on reproducible research computing. The month before, she had been awarded a 2016 Leamer-Rosenthal Prize, which celebrates those "working to forward the values of openness and transparency in research". In this Q&A, she talks flying snakes, 'repro-packs' and copyright.
go.nature.com/2y2vacg

The sound of DNA
With an alphabet comprising just four letters, a DNA sequence isn't much to look at. So when sequence-analysis tools want to highlight key elements, they typically do so using colour or font, or by overlaying other types of information. In the not-too-distant future, there may be another option. Molecular biologist and part-time drummer Mark Temple at Western Sydney University, Australia, describes DNA sonification, "an auditory display tool" for DNA: sequence in, audio out. "I'm not saying audio by itself is the bees' knees for interpreting DNA sequence," Temple says, "but surely audio can contribute to your visual interpretation."
go.nature.com/2fadzxi

in Singapore found that about 40% of the 34,544 most-popular open-source projects hosted on the coding collaboration site GitHub used continuous integration in some form.

Only a few of those open-source projects might be considered scientific software, but an increasing number of scientists are looking to continuous integration to automate all sorts of time-consuming tasks, from testing code to updating documents with the latest data.

Researchers at institutions such as CERN, Europe's particle-accelerator laboratory near Geneva, Switzerland; the Pacific Northwest National Laboratory in Richland, Washington; and the Ontario Institute for Cancer Research in Toronto, Canada, have embraced the practice, but adoption in the scientific sector remains relatively sparse.

For Neubert, continuous integration ensures that the pipeline's behaviour remains correct and consistent as his team refines its code, providing an "incredibly valuable" safeguard. "There is a real danger of just missing something or making a slight mistake," he says.

EXCEPTIONS

A variety of continuous integration services exist. These include the open-source Drone, and commercial options such as CircleCI, Codeship, GitLab, Shippable and Travis CI, all of which offer pricing tiers based on the desired testing behaviour, number of users and whether the project is public or private. Travis CI, for instance, is free for open-source projects; private projects cost from US$69 per month. Shippable offers a free basic service for public projects, but charges $25–150 per month for support for private projects and greater computing power, among other features.

Researchers should consider what is a suitable and worthwhile investment, however. Not every project needs continuous integration and setting up and configuring a service can be challenging. Further difficulties can arise if the services need to interact with software or data with legal restrictions on its use, says Daniel Himmelstein, a data-science postdoc at the University of Pennsylvania in Philadelphia.

Also, code is often used only once, making the cost even less worthwhile. "For day-to-day research coding, the amount of code is not large enough to make continuous integration valuable," says Andrea Zonca, a specialist in high-performance computing at the University of California, San Diego. He uses Travis CI when publishing code, but most that he writes is for his own one-time use and is not executed again.

Computing costs can also mount if code is being constantly updated and requires repeated testing, which is why Neubert's lab only tests its most critical data analyses after code changes.

Despite these challenges, continuous integration services tend to improve code quality, says Björn Grüning, a bioinformatician at the University of Freiburg in Germany, especially on large projects such as Galaxy, a bioinformatics toolkit that Grüning, along with about 160 others, contributes to.

According to Grüning, continuous integration has shortened the turnaround time for approving contributions to the Galaxy project and given programmers more confidence when submitting new features and fixes. Before these services were available, it was often impractical for researchers in such projects to test every new feature collaborators proposed because they didn't have the time, he says.

Some researchers use continuous integration to automate non-programming tasks. In April, as part of a project studying how ecosystems change over time, Ethan White, an ecologist at the University of Florida in Gainesville, helped to configure Travis CI to update tables and plots automatically with new field or weather-station data, saving the research team up to 5 hours a month.

"Continuous integration has shortened the turnaround time for approving contributions."

Continuous integration helps Himmelstein automate revisions to scientific papers, citations and web pages following text or code updates. Without continuous integration, he says, human maintainers would probably "get lazy and update the manuscript less frequently than every change".

INITIALIZING

Whether hosted externally by a third party or on a user's own machine, the continuous integration service is controlled with a custom set of instructions. This configuration file defines the tasks to be run and sets up the server with the correct environment — the operating system and software libraries — required to run them. The service then executes those instructions at set times or on receipt of a code or data update.

University of Pennsylvania bioinformatician Casey Greene, who uses continuous integration to rerun his data analyses, has tested many of today's most popular services. "The good news about all of these services is that they're quite similar," he says.

Subtle differences do exist, for instance in the number of concurrent jobs users can run, or the amount of computing power available to run them. "I'd encourage people to dig into the limits of each service to make sure they are compatible with their workflows," advises Greene.

Although continuous integration adoption in science right now is small, it is growing, and more researchers should get on board, Greene says. Getting up to speed takes time, he acknowledges, but often, the effort is worth the reward. "Scientists analysing data should have it in their toolbox." ■

Andrew Silver *is a science and technology writer in London.*