# Insprint automation

Enter **in-sprint** test **automation**, an ideal development scenario where the entire testing process — from creation and implementation to execution and reporting — happens in one sprint. To many, **in-sprint automation** is a unicorn or leprechaun.

# 5 steps of successful Insprint automation

## **Developers must check-in code early and often within a sprint**

This is the first and most important tactic that makes in-sprint automation possible. It's also the most overlooked. Leaders and team members tend to only focus on testing once it’s become a bottleneck—a symptom of the problem—rather than look at the root problem—developers failing to check in.

Developers often wait until sprint end to check in completed features because they don't want to affect other people too early and they don't want to get blamed for breaking the build. But this leads to other problems such as losing work, not being able to go back to previous versions, and not providing ample time for testing. If testers barely have enough time to do manual testing how can they possibly be expected to write automated tests?

This is why it is so imperative that developers start deploying code to the test environment early and often within the sprint. They should not wait until the feature is complete, but rather until it’s complete enough for testers to have something to work with—and then to continually deploy as more and more of the feature is coded. By doing this, testers can begin to write automated test components, and continually build the components and tests as more of the feature is deployed.

## **Automation engineers must be on development scrum teams**

A lot of times automation engineers sit on a separate team where they work through a regression suite backlog. Instead, they should be placed directly on a development scrum team where they can advocate for the best ways to perform test automation and thereby influence the team to implement in-sprint automation.

What’s more, by partnering automation engineers with developers, it sends a clear message that test automation is a primary focus of the sprint and not an afterthought.

## **Automation engineers must be well trained**

It might seem like a no-brainer, but it’s a frequent miss among scrum teams. Beyond just making sure that automation engineers actively participate as team members, it’s important that they understand *all*of the requirements for automation to succeed. This means that they must be highly skilled in*both*testing processes as well as code writing.

Think of automation engineers as multi-function individuals—they should be capable of writing and executing manual test cases as well as automated tests in the team’s programming language of choice. A standout automation engineer is also extremely nimble, able to quickly write new tests along with all corresponding automation features while keeping up with the quick pace of agile development.

Where many organizations fail in this regard is when they attempt to convert manual testers to automation engineers. Manual testers often do not have the necessary programming skills to effectively write new tests and testing components. Without being able to write new tests quickly and shortly after features are deployed to the testing environment, in-sprint automation is not possible.

## **Written test cases should be small, concise, and descriptive**

At all costs, testers should refrain from writing long workflows that verify multiple components at once. Instead, a good automated test case should verify only one feature or sub-feature at a time and should be very explicit as to what steps the test is completing.

By writing clearly defined steps and expected results, the writing process is shortened and the automation engineer can focus on the actual automation rather than wasting time trying to figure out what the test is set to verify.

As you can imagine, it’s important that the automation engineer is closely involved in the test-case process, whether that means they’re writing test cases or simply talking with other team members to determine what is being verified within the sprint. This way, they’ll know how to best write the automation.

## **Add automated testing to the Definition of Done**

The Definition of Done (DOD) ensures that everyone on the scrum team knows exactly what is expected of what the team is supposed to deliver. By adding automation testing as a DOD item, the team gains agreement on the importance of automated testing processes and can make it a primary focus.

Another benefit of including automation as a DOD item is that it will continue to be a topic of discussion in post-sprint meetings as team members reflect on what went well and what could be improved. This way, they can continually improve the process until in-sprint automation is achieved.

[](https://www.google.com/search?q=insprint+automation+challenges&hl=en&sxsrf=ALeKk01t4_3_NQmU1cAfm8LEKQDEOShfhw:1614411941055&tbm=isch&source=iu&ictx=1&fir=oGHf-d8aLxWOlM%252C3XWdZGpVLM0HiM%252C_&vet=1&usg=AI4_-kQgeJ_Sp-kNK_Da8xC-8dqETnTgDg&sa=X&ved=2ahUKEwjF2pGSyYnvAhXT4zgGHQM6B28Q9QF6BAgSEAE" \l "imgrc=oGHf-d8aLxWOlM)

# **In-Sprint Test Automation Challenges for Regression Testing**

* Lack of time to **automate** in short sprints.
* Lack of a unified continuous testing platform.
* Flexibility to enhance the tool/framework.
* Availability of the right set of tools.
* Teams' approach within the sprint.