Agile Advocate: The Developer’s Sandbox

An isolated environment allows iterative development, experimentation and fast feedback.

Bill Drew
Fri, 01/19/2018 - 12:26

Illustration: akindo/iStock

Introduction
Why Do We Need CI Again?

Just to review, the primary purpose of the continuous integration process, commonly referred to as the “build,” is to support continual integration of changes to the code base from different developers and teams into a viable release candidate. This usually involves numerous phases for building multiple components and services to bring together a functioning release candidate that product owners can interrogate and testers can validate.

And it’s imperative you do not break the CI process! And if you do, fixing it should be your new No. 1 priority, which is a well-known and accepted best practice in the industry. And it’s reasonable to expect because the CI process maintains the integrity of the code base. It ensures developers and teams are integrated and in sync. And just as importantly, it’s the factory that provides the release candidates to product owners and testers to maintain the feedback loop with developers.

Commit Early and Often

Agile and industry best practices call for developers to commit changes to the code base regularly to keep changes to the code base small and more manageable. If changes must be rolled back or course corrections must be made to the committed code, the impact and risk is reduced with these smaller changes. Also, continual and fine-grained changes tend to lessen the integration burden between developers and teams.

However, these constant changes to the code base result in more builds. And without an alternative, developers begin using the build and CI process to test these changes and to integrate their code. Using the CI process to check code compliance
or validate experimentation understandably results in more broken builds and lessens the integrity of the CI process.

So, how do we satisfy these seemingly incompatible requirements? On one hand, we need to maintain a stable build while on the other hand, we insist on constant changes to the code base that tends to destabilize the build.

**Welcome to the Sandbox**

What developers need is an isolated environment in which they can practice iterative development, experiment with new ideas and continue to get fast feedback without relying on the full CI process or negatively impacting the development cadence and feedback loop.

The developer sandbox is designed to serve just this purpose. It is not a completely new CI, build or build framework. The developer sandbox uses the same build initiated by the CI server when a commit is made to the code base. But instead of a full build — required to create a release candidate — a local build is run on a developer’s workstation that can execute just a fraction of the build the full CI process performs.

This local build focuses on executing the phases of the larger CI most important to a developer, such as static code analysis, compilation and unit and component testing. It may even go beyond those steps to support some level of integration testing. But even with these additional steps, the feedback on this local build is far quicker than expected from the full CI process. This facilitates experimentation and test-driven development and does not rely on the full CI process to validate ideas.

While using the sandbox, a developer must be cognizant of changes to the code, paying particular attention to notification of code changes that may impact the developer’s work. Any relevant changes to the code base should be pulled down into their local repository to incorporate it into local builds, which ensures a developer’s local work is still aligned with ongoing work on the code base.

And before committing any changes to the repository, developers should ensure their local version contains all the relevant updates from the repository. Before committing any changes, a developer would most likely execute a more complete build locally, which may include integration and application programming interface
testing, to ensure the changes integrate successfully, once committed. At this point, a developer should have a high degree of confidence that when he or she commits changes, the CI will successfully complete.

The Workflow

Figure 1 provides a diagram and step-by-step workflow of how developers would use a local sandbox and their interactions as each commits to the Version Control System to trigger the CI process.

1. A local build is initiated by a developer

2. After checking that the local repository is up to date, a commit is made which
3. Upon receiving notifications of the commit and successful build, the other developers pull down the change to their local repository.

4. They both initiate a local build to incorporate the changes and confirm their local builds still work against their changes.

5. A second commit is made, triggering the CI process and sending notifications of the successful build and commits to the other developers.

6. This time only, one of the developers decides to update his or her local repository.

7. The developer executes the local build using updates from the shared repository.

**Worthwhile Investment**

The full CI can be a time-consuming, resource-intensive process whose integrity and output must be maintained. The sandbox provides the developer an alternative to the full, heavyweight CI with a fast-fail environment that supports a high-development cadence. Using a local sandbox encourages test-driven development, testing and experimentation, and in the end, greatly reduces the chances of a broken CI process and helps maintain its integrity.

It will take some additional time and resources to establish a developer sandbox. It will also require changes in developer habits and expectations. But I think you’ll find that supporting TDD and experimentation while safeguarding the integrity of the CI process and maintaining the feedback loop between product owners and the development teams is well worth the investment.

--- Sign up for our weekly newsletter to receive the latest analysis and insights on emerging federal technologies and IT modernization.

View printer friendly version