# The Landscape

Continuous Integration systems provide the benefits of automated and simplified testing for developers. These tests can be executed and information on these tests can be reported back faster and more reliably.

When observing the landscape of Continuous Integration systems, the landscape primarily focuses on the interactions and integrations between the client and central server. In the examples provided, the central server, often referred to as the *build master* has the job of scheduling and organizing the multiple client instances. The clients, referred to as the *build slaves*, receive these instructions from the central server and perform and execute the central server's detailed instructions. In another example, it is mentioned how the client can execute the central server's instructions and deposit the information in a separate external server. The final example, a hybrid of the two, mentioned how builds can be ordered to run independently and have results sent back to the central server, or a set of builds can be set to execute, test, and report back to the central server. Solid internal communication between a central server and the clients are part of the core fundamentals for a well-working Continuous Integration system.

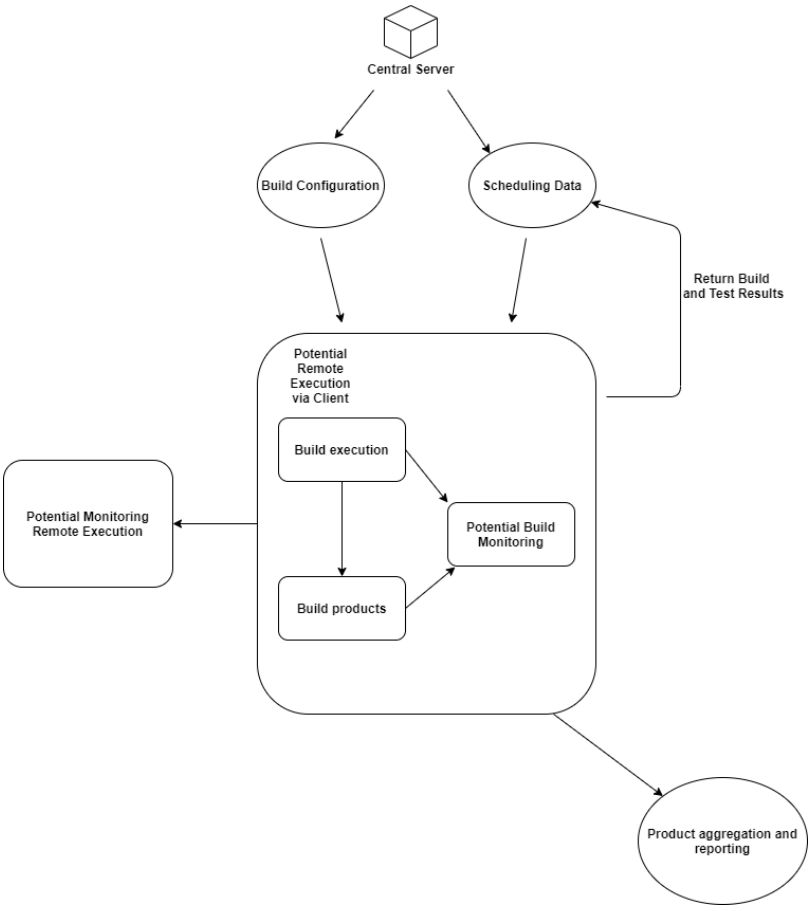


Fig 1: Internal communications of a continuous integration system with potential before and after build monitoring.

The core functionalities of a continuous integration system appear to be very simplistic. Software should be built, run, tested, and these test results should be reported. Continuous Integration systems built in the real world carry out more than just these basic tasks. There is the potential for these systems to have other functionalities such as releasing packages and creating and releasing progress reports. transferring build products to a central repository, or the functionality to create and release long-term progress reports. These different functionalities when utilized with a CI system can greatly benefit the developers by providing them with the ability to receive, push, and manage any of the builds and the information retrieved from these builds.

With all the communication between the central server and the clients, it is easy to forget that these continuous integration systems also communicate with other systems. These interactions between systems could be any form of contact or passing of information between the CI system and the external systems. Examples of these interactions could be a notification on if a build fails, to something like an external build request.

This described landscape of Continuous Integration systems works as a solid foundation for the real-world implementations to build and branch off of. The mentioned examples show how each CI system can be different from one another strictly due to the features and methods they choose to utilize. Each of the mentioned examples displays their own set of benefits, uniquely tailored to the functionalities they've chosen to deploy. Ultimately it is up to the developers to determine which architecture and which set of features best suits their project and overall goal when utilizing Continuous Integration.