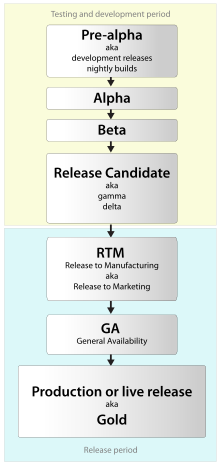
# Continuous Integration

## Software Release Life Cycle

A software release life cycle is the sum of standardized stages a software undergoes from its initial development to its eventual release. At each stage the software obtains a particular leve of stability.

### Software Release Life Cycle map



### Pre-alpha

Pre-alpha refers to all activities performed during the software project before formal testing. These activities can include requirements analysis, software design, software development, and unit testing.

### Alpha

The alpha phase of the release life cycle is the first phase to begin software testing (alpha is the first letter of the Greek alphabet, used as the number 1). In this phase, developers generally test the software using white-box techniques. Additional validation is then performed using black-box or gray-box techniques, by another testing team. Moving to black-box testing inside the organization is known as alpha release.

Alpha software can be unstable and could cause crashes or data loss. Alpha software may not contain all of the features that are planned for the final version.

### Beta

"Beta Test" redirects here. For the film, see Beta Test (film).

Beta, named after the second letter of the Greek alphabet, is the software development phase following alpha. Software in the beta stage is also known as betaware.[3] Beta phase generally begins when the software is feature complete but likely to contain a number of known or unknown bugs.[4] Software in the beta phase will generally have many more bugs in it than completed software, as well as speed/performance issues and may still cause crashes or data loss. The focus of beta testing is reducing impacts to users, often incorporating usability testing. The process of delivering a beta version to the users is called beta release and this is typically the first time that the software is available outside of the organization that developed it. Beta version software is often useful for demonstrations and previews within an organization and to prospective customers. Some developers refer to this stage as a preview, preview release, prototype, technical preview / technology preview (TP),[5] or early access. Some software is kept in perpetual beta, where new features and functionality are continually added to the software without establishing a final "stable" release.

Beta testers are people who actively report issues of beta software. They are usually customers or representatives of prospective customers of the organization that develops the software.

### Release candidate

A release candidate (RC), also known as "going silver", is a beta version with potential to be a final product, which is ready to release unless significant bugs emerge. In this stage of product stabilization, all product features have been designed, coded and tested through one or more beta cycles with no known showstopper-class bugs. A release is called code complete when the development team agrees that no entirely new source code will be added to this release. There could still be source code changes to fix defects, changes to documentation and data files, and peripheral code for test cases or utilities.

### Release

Once released, the software is generally known as a "stable release".

### Release to manufacturing (RTM)

The term "release to manufacturing", also known as "going gold", is a term used when a software product is ready to be delivered. This build may be digitally signed, allowing the end user to verify the integrity and authenticity of the software purchase. A copy of the RTM build known as the "gold master" or GM is sent for mass duplication if applicable.

### General availability (GA)

Milestones in a product life cycle: general availability (GA), end of life announcement (EOLA), last order date (LOD), and end-of-life (EOL)

General availability (GA) is the marketing stage at which all necessary commercialization activities have been completed and a software product is available for purchase. Commercialization activities could include security and compliance tests, as well as localization and worldwide availability.

### Are You Really Doing CI?

#### Do you check in to mainline once per day?

You need to make sure your code integrates. If you don’t check your code together with everyone else’s changes frequently, you end up making future integration harder. Even if you are using short-lived branches to manage changes, integrate as frequently as you can into a single mainline branch.

#### Do you have a suite of tests to validate your changes?

Without tests, we just know that syntactically our integration has worked, but we don’t know if we have broken the behavior of the system. CI without some verification that our code behaves as expected isn’t CI.

#### When the build is broken, is it the #1 priority of the team to fix it?

A red build means the last change possibly did not integrate.

You need to stop all further check-ins that aren’t involved in fixing the builds to get it passing again. If you let more changes pile up, the time it takes to fix the build will increase drastically. I’ve worked with teams where the build has been broken for days, resulting in substantial efforts to eventually get

a passing build.