CI/CD

GIVE YOUR APPLICATION AUTO-DEPLOY SUPERPOWERS

INTRODUCTION

CONTINUOUS INTEGRATION (CI) AND CONTINUOUS DEPLOYMENT (CD) ARE POPULAR SOFTWARE DEVELOPMENT

PRACTICES FOR AUTOMATION AND SHORTENING FEEDBACK TIMES. HOWEVER, SETUP IMPROPERLY, YOUR CI/CD

PIPELINES COULD INSTEAD CAUSE DELAYS IN DEVELOPMENT.

FOR THAT REASON, REVIEW THESE CI/CD BEST PRACTICES TO ENSURE THAT YOUR PIPELINES ARE EFFECTIVE

AND EFFICIENT.

WHAT IS CONTINUOUS INTEGRATION?

The processes used by your team have a direct impact on the efficiency of your software development workflow.
For that reason, it is important for your team to adopt processes that streamline your software development — like Continuous Integration.

WHAT ARE THE BENEFITS OF CONTINUOUS INTEGRATION?

BY IMPLEMENTING CONTINUOUS INTEGRATION, SOFTWARE DEVELOPMENT TEAMS BENEFIT FROM:

EASIER BUG FIXES
Identifying issues sooner makes it easier for developers to fix errors, vulnerabilities, and defects in the code. What's more, this helps to ensure that an issue will be fixed correctly, resulting in a build that's issue free and working as quickly as possible.

REDUCED PROJECT RISK
Encouraging small, modular changes to the code enables new functionality to be backed out of a release more quickly, or even prevented from entering the main code stream altogether. This minimizes the impact on other developers.

IMPROVED SOFTWARE QUALITY
Maximizing the value of CI means detecting as many issues as possible in each integration build, through automation. This increases the breadth, depth, and repeatability of the tests while avoiding manual testing.

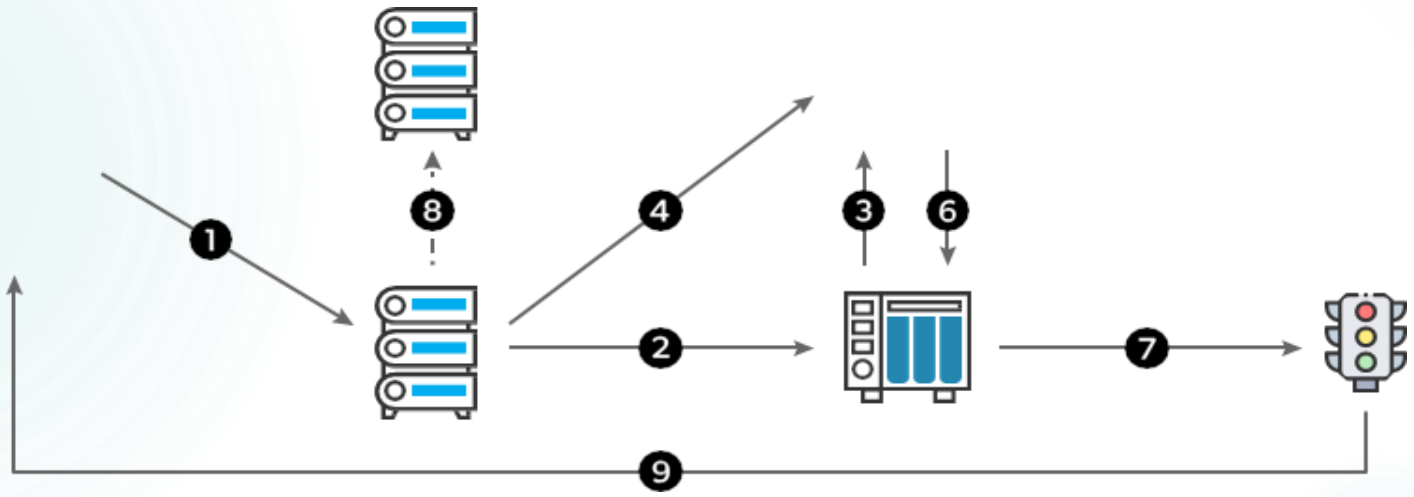
HIGHER PRODUCTIVITY
Automating these tasks frees up developers to focus on higher-value feature development.

WHAT IS THE DIFFERENCE BETWEEN CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY?

Continuous Integration (CI) and Continuous Delivery (CD) are both software development practices.

CI is used during the build and test phase. CD is used once changes are committed. The ultimate aim of CD is to always have validated and verified code in the code repository — or version control system — ready for release.

9 STEPS OF CONTINUOUS INTEGRATION



1. Developers check code into the version control system's staging repository.
2. The Version Control System (VCS) or code repository notifies the CI server that a commit has occurred. Or, the CI server polls the repository periodically looking for commits.
3. The CI server starts the build process on a build server.
4. The code containing the latest commit — ideally just the minimum file set — is checked out of the repository into a local workspace on the build server.
5. The changed code is built, analyzed, and tested.
6. Important results are reported back to the CI server, along with any important details and files that need to be retained.
7. The CI server sets the final — Pass / Fail — result of the build.
8. If the build met the success criteria, then the committed change may proceed through the development cycle - transferred to the real repository or merged to the main development stream. If the build failed, the committed changes are blocked from proceeding until those issues are resolved.
9. The CI server notifies any parties who have registered interest in the build. They can then log into the CI server to view the status plus any additional information.

THANK YOU

ZIAD MOHAMED ALI

ZMA.CEE@GMAIL.COM