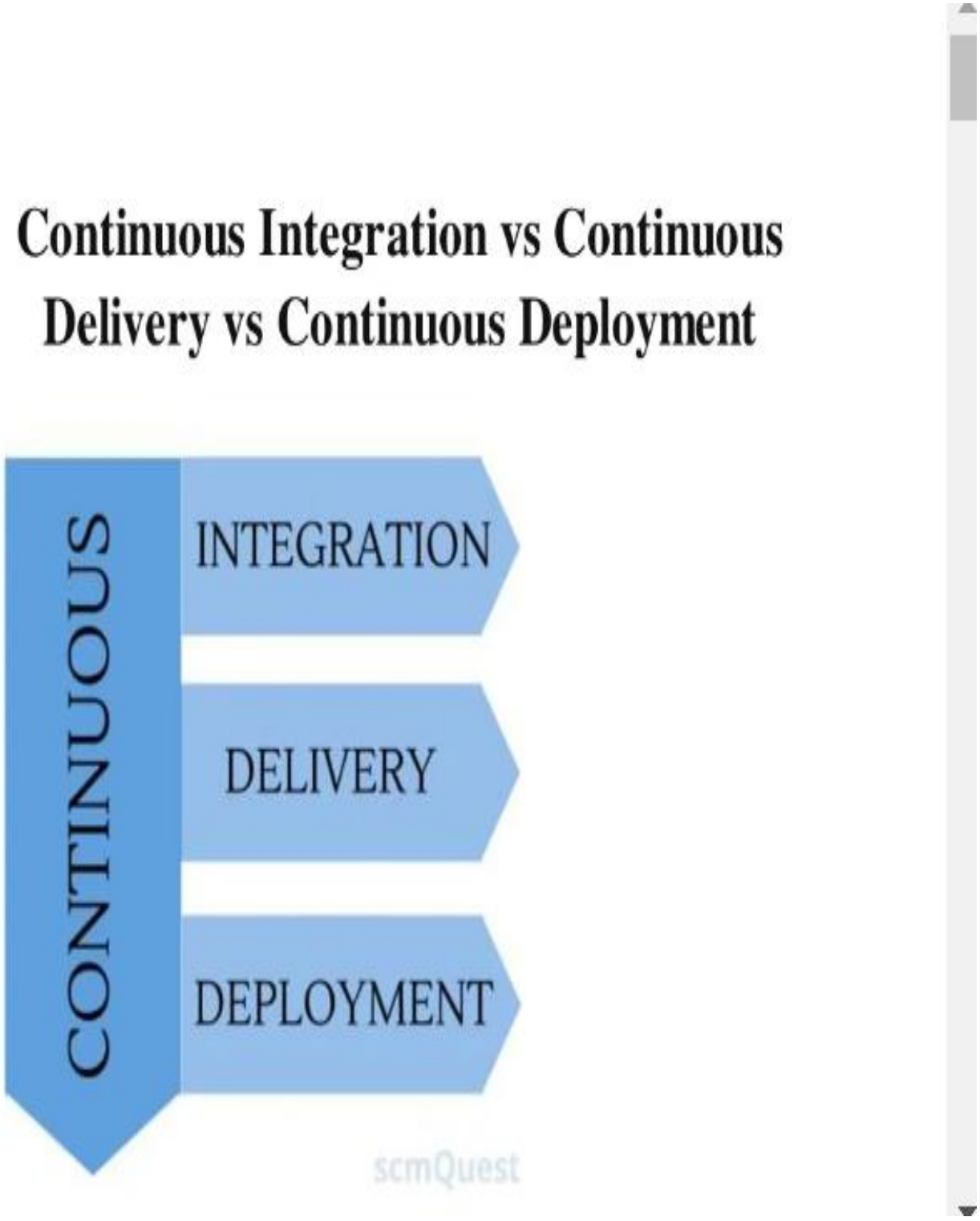

Continuous Integration vs Continuous Delivery vs Continuous Deployment
CONTINUOUS
INTEGRATION
DELIVERY
DEPLOYMENT
scmQuest

What is CI/CD?

Continuous integration (CI) and continuous delivery (CD) are the processes that are used to build, package, and deploy your application. Basically, it lays out some practices to follow in order for the code you write to more quickly and safely get to your users and ultimately generate value.

Continuous Integration (CI) is a development practice that helps ensure that software components work together. CI allows you to continuously integrate code into a single shared and easy to access repository.

Continuous delivery (CD) is the ability to deploy your integrated code into production without the need of human intervention. CD allows you to take the code stored in the repository and continuously delivery it to production.

Why CI/CD Matters

CI/CD practices should matter to you as it helps get continuous feedback not only from your customers but also from your own team. Moreover, in an organization, it can also lead to big advantages. Some of the notable benefits of implementing CI/CD pipelines to your everyday software development process are:

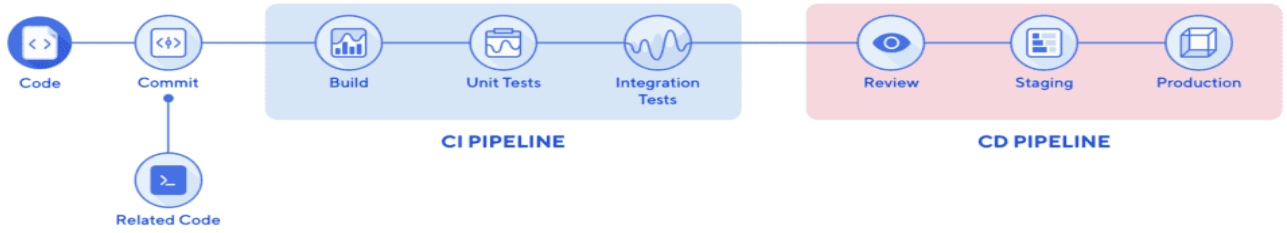
- **Reduce costs:** Using automation in the CI/CD pipeline helps reduce the number of errors that can take place in the many repetitive steps of CI and CD.
- **Smaller code changes:** One technical advantage of CI and CD is that it allows you to integrate small pieces of code at one time. This helps developers to recognize a problem before too much work is completed afterward.
- **Faster release rate:** Failures are detected faster and as such, can be repaired faster, leading to increasing release rates.
- **Fault isolations:** Designing your system with CI/CD ensures that fault isolations are faster to detect and easier to implement.
- **More test reliability:** Using CI/CD, test reliability improves due to the bite-site and specific changes introduced to the system, allowing for more accurate positive and negative tests to be conducted.

All benefit are as outline simply below,

Technical Language	Value	Translation
Catch Compile Errors After Merge	Reduce Cost	Less developer time on issues from new developer code
Catch Unit Test Failures	Avoid Cost	Less bugs in production and less time in testing
Detect Security Vulnerabilities	Avoid Cost	Prevent embarrassing or costly security holes
Automate Infrastructure Creation	Avoid Cost	Less human error, Faster deployments
Automate Infrastructure Cleanup	Reduce Cost	Less infrastructure costs from unused resources
Faster and More Frequent Production Deployments	Increase Revenue	New value-generating features released more quickly

Technical Language	Value	Translation
Deploy to Production Without Manual Checks	Increase Revenue	Less time to market
Automated Smoke Tests	Protect Revenue	Reduced downtime from a deploy-related crash or major bug
Automated Rollback Triggered by Job Failure	Protect Revenue	Quick undo to return production to working state

CI/CD Pipeline



A CI/CD pipeline is a path for delivering a unit of change that starts from development to delivery. It helps you automate steps in your software delivery process and allows application development teams to release software quickly.

A CI/CD pipeline workflow usually consists of the following discrete steps:

Phase 1: Commit

When developers complete a change, they commit the change to the repository.

Phase 2: Build

Source code from the repository is integrated into a build.

Phase 3: Automate tests

Automated tests are run against the build. Test automation is an essential element of any CI/CD pipeline.

Phase 4: Deploy

The built version is delivered to production.

Importance of Test Automation for CI/CD

Test automation is a crucial component of any CI/CD pipeline. The benefits of applying CI/CD cannot be realized if there is a lack of automated testing and a low level of test coverage. Teams need to perform automated testing at all levels including unit, integration, and system testing.

Also, teams need to apply automation testing for multiple testing types such as functionality, usability, performance, load, stress, and security.

Conclusion

DevOps has brought with it the rise of CI/CD. In the present context, CI/CD practices are the most widely accepted choice to shorten software development and delivery cycle time. Many software tools are available to support implementing CI/CD practices. Katalon Studio provides a comprehensive set of features for API, Web, and mobile testing. It can be easily integrated into a CI/CD pipeline to handle unit (API services), integration, and system testing.