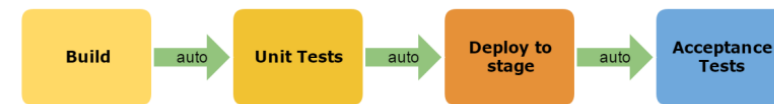# UDAPEOPLE with CI/CD Pipeline

**Done By: Raghda Mahgoub**

# What is CI/CD Pipeline And Why Do We Need it

- **Continuous Integration (CI)** allows you to continuously integrate code into a single shared and easy to access repository.

- **Continuous Delivery (CD)** allows you to take the code stored in the repository and continuously deliver it to production.

- **Continuous Deployment** is a step up from Continuous Delivery in which every change in the source code is deployed to production automatically, without explicit approval from a developer. A developer's job typically ends at reviewing a pull request from a teammate and merging it to the master branch. A CI/CD service takes over from there by running all tests and deploying the code to production, while keeping the team informed about outcome of every important event.

- **Continuous deployment** requires a highly developed culture of monitoring, being on call, and having the capacity to recover quickly

- **CI/CD** creates a fast and effective process of getting your product to market before your competition as well as releasing new features and bug fixes to keep customers happy.

**Continuous Integration**

Build → auto → Unit Tests → auto → Deploy to stage → auto → Acceptance Tests

**Continuous Delivery**

Build → auto → Unit Tests → auto → Deploy to stage → auto → Acceptance Tests → manual → Deploy to production

**Continuous Deployment**

Build → auto → Unit Tests → auto → Deploy to stage → auto → Acceptance Tests → auto → Deploy to production

# Benefits of CI/CD

### 1. Smaller Code Changes

❖ One technical advantage of continuous integration and continuous delivery is that it allows you to integrate small pieces of code at one time.

❖ Using continuous testing, these small pieces can be tested as soon as they are integrated into the code repository, allowing developers to recognize a problem before too much work is completed afterward. This works really well for large development teams who work remotely as well as those in-house as communication between team members can be challenging.

### 2. Fault Isolations

❖ Fault isolation refers to the practice of designing systems such that when an error occurs, the negative outcomes are limited in scope. Limiting the scope of problems reduces the potential for damage and makes systems easier to maintain.

❖ Designing your system with CI/CD ensures that fault isolations are faster to detect and easier to implement. Fault isolations combine monitoring the system, identifying when the fault occurred, and triggering its location. Thus, the consequences of bugs appearing in the application are limited in scope. Sudden breakdowns and other critical issues can be prevented from occurring with the ability to isolate the problem before it can cause damage to the entire system.

### 3. Faster Mean Time To Resolution (MTTR)

❖ MTTR measures the maintainability of repairable features and sets the average time to repair a broken feature. Basically

❖ CI/CD reduces the MTTR because the code changes are smaller and fault isolations are easier to detect.

### 4. More Test Reliability

❖ Using CI/CD, test reliability improves due to the bite-size and specific changes introduced to the system, allowing for more accurate positive and negative tests to be conducted. Test reliability within CI/CD can also be considered Continuous Reliability. With the continuous merging and releasing of new products and features, knowing that quality was top of mind throughout the entire process assures stakeholders their investment is worthwhile.

### 5. Faster Release Rate

❖ Failures are detected faster and as such, can be repaired faster, leading to increasing release rates. However, frequent releases are possible only if the code is developed in a continuously moving system

# Benefits of CI/CD

### 6. Smaller Backlog

❖ Incorporating CI/CD into your organization's development process reduces the number of non-critical defects in your backlog. These small defects are detected prior to production and fixed before being released to end-users.

❖ The benefits of solving non-critical issues ahead-of-time are many. For example, your developers have more time to focus on larger problems or improving the system and your testers can focus less on small problems so they can find larger problems before being released. Another benefit (and perhaps the best one) is keeping your customers happy by preventing them from finding many errors in your product.

### 7. Customer Satisfaction

❖ The advantages of CI/CD do not only fall into the technical aspect but also in an organization scope. The first few moments of a new customer trying out your product is a make-or-break-it moment.

❖ Keep your customers happy with fast turnaround of new features and bug fixes. Utilizing a CI/CD approach also keeps your product up-to-date with the latest technology and allows you to gain new customers who will select you over the competition through word-of-mouth and positive reviews.

❖ customers are the main users of your product. As such, what they have to say should be taken into high consideration. Whether the comments are positive or negative, customer feedback and involvement leads to usability improvements and overall customer satisfaction.

### 8. Increase Team Transparency and Accountability

❖ CI/CD is a great way to get continuous feedback not only from our customers but also from your own team..

❖ CI is mostly focused on the development team, so the feedback from this part of the pipeline affects build failures, merging problems, architectural setbacks, etc. CD focuses more on getting the product quickly to the end-users to get the much-needed customer feedback. Both CI and CD provide rapid feedback, allowing you to steadily and continuously make your product even better.
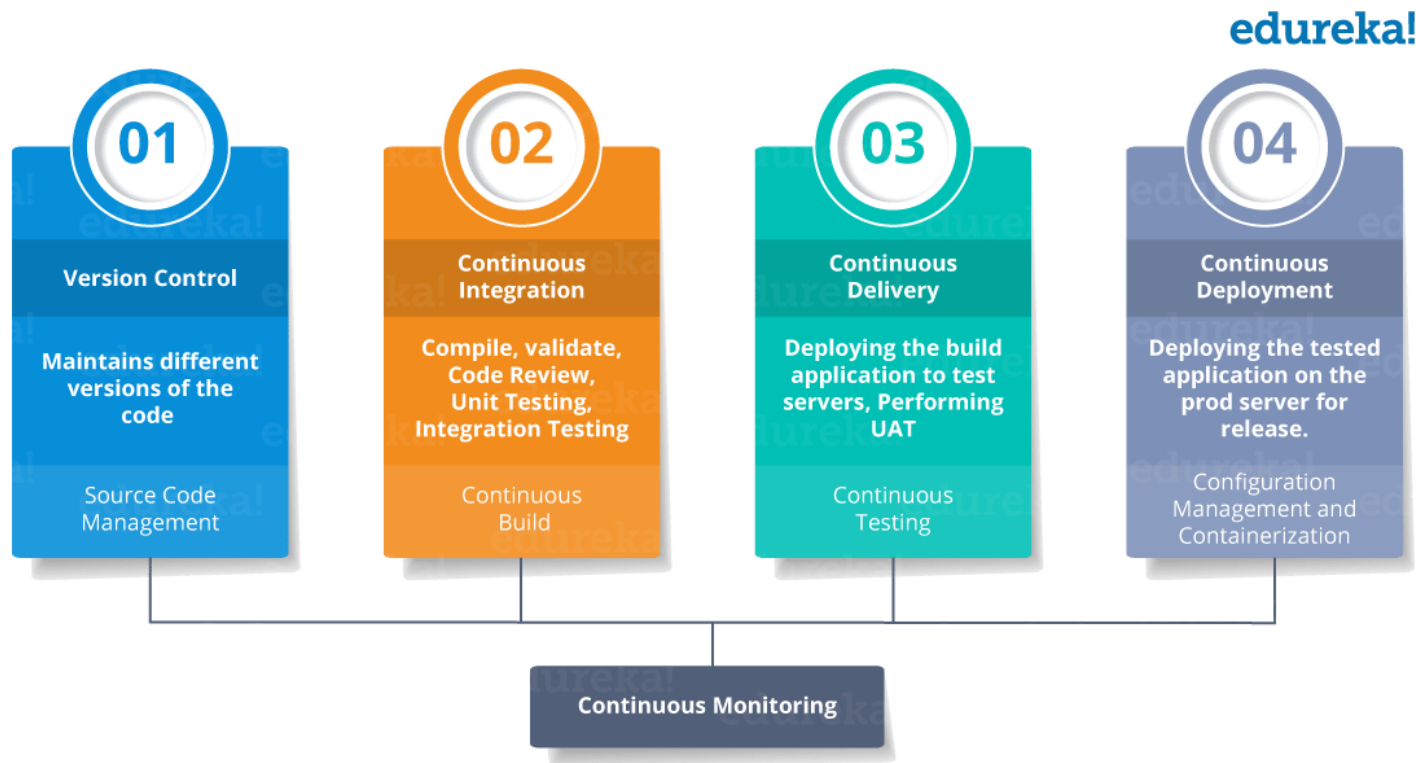
### 9. Reduce Costs

❖ Automation in the CI/CD pipeline reduces the number of errors that can take place in the many repetitive steps of CI and CD. Doing so also frees up developer time that could be spent on product development as there aren't as many code changes to fix down the road if the error is caught quickly.

### 10. Easy Maintenance and Updates

❖ Maintenance and updates are a crucial part of making a great product.

❖ Make sure the pipeline runs smoothly by incorporating when to make changes and releases. A great way to ensure maintenance doesn't affect the entire system is to create microservices in your code architecture so that only one area of the system is taken down at one time.

➢ There are many tools that can help enable a smoother transition to a CI/CD process.

➢ Testing is a large part of that process because even if you are able to make your integrations and delivery faster, it would mean nothing if was done so without quality in mind.

➢ The more steps of the CI/CD pipeline that can be automated, the faster quality releases can be accomplished.

# Conclusion

edureka!

**01**

**Version Control**

**Maintains different versions of the code**

Source Code Management

**02**

**Continuous Integration**

**Compile, validate, Code Review, Unit Testing, Integration Testing**

Continuous Build

**03**

**Continuous Delivery**

**Deploying the build application to test servers, Performing UAT**

Continuous Testing

**04**

**Continuous Deployment**

**Deploying the tested application on the prod server for release.**

Configuration Management and Containerization

**Continuous Monitoring**