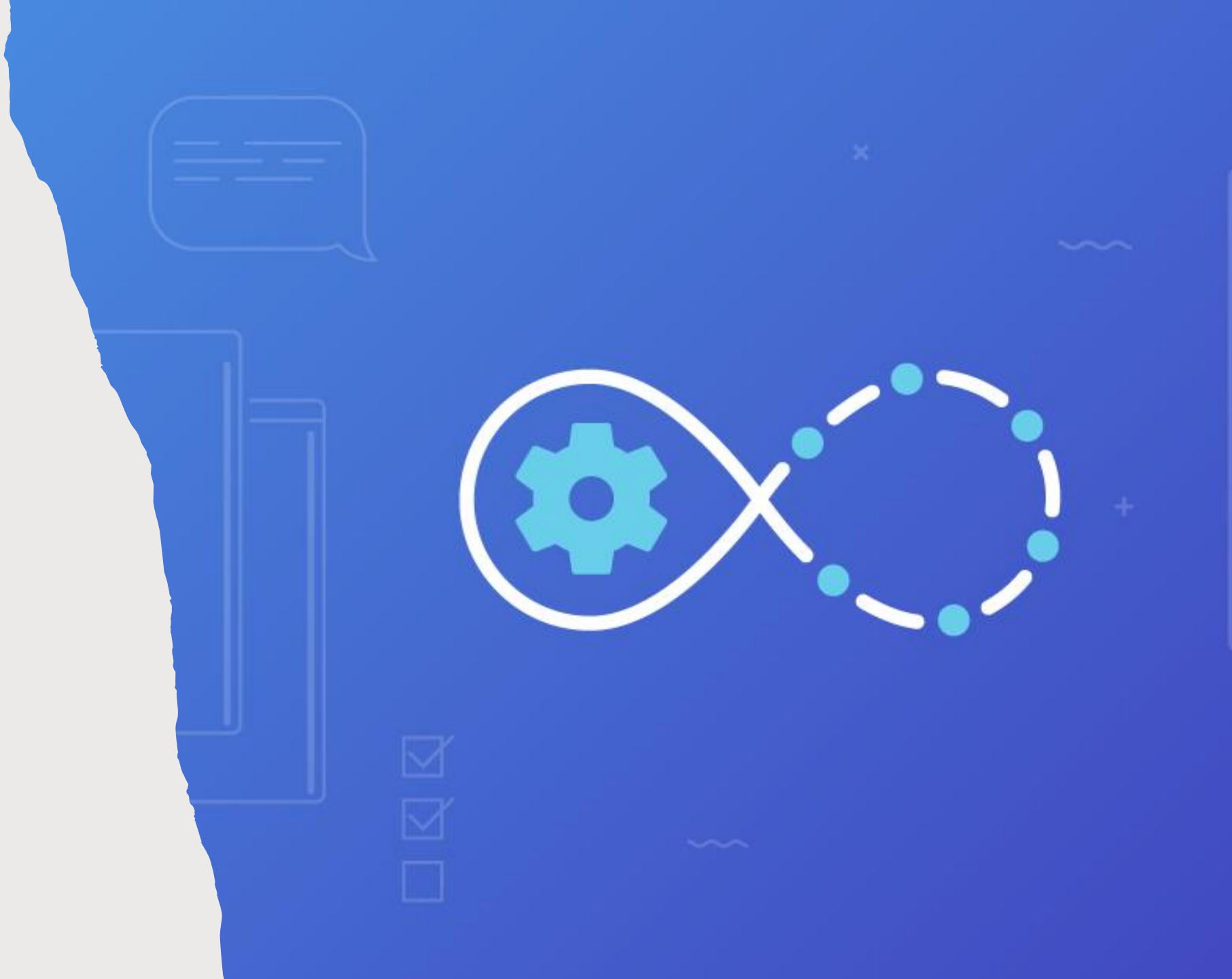
CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY

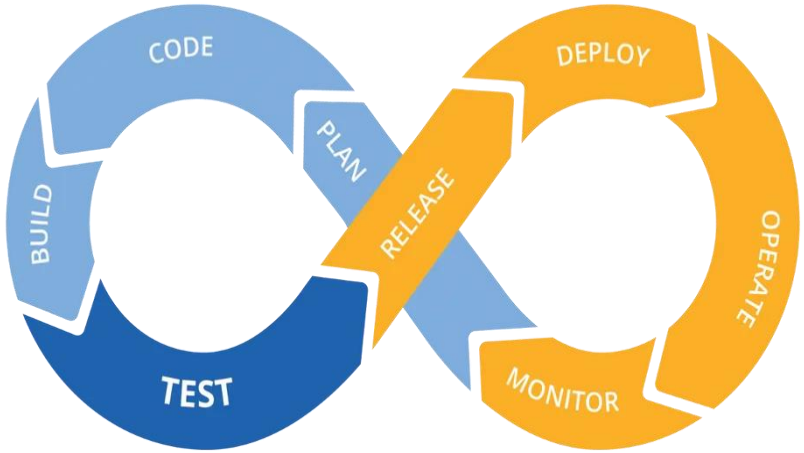
CI/CD

Ahmed Adel



WHAT IS CI/CD

- **Continuous integration (CI) and continuous delivery (CD)**, also known as CI/CD, embodies a culture, operating principles, and a set of practices that application development teams use to deliver code changes more frequently and reliably.
- **Continuous integration (CI)** is *The practice of merging all developers' working copies to a shared mainline several times a day. It's the process of "Making". Everything related to the code fits here, and it all culminates in the ultimate goal of CI: a high quality, deployable artifact!*
- **Continuous Deployment** is *A software engineering approach in which the value is delivered frequently through automated deployments. Everything related to deploying the artifact fits here. It's the process of "Moving" the artifact from the shelf to the spotlight*



WHAT ARE THE BENEFITS OF CI/CD?

- **Faster time to market**

The primary goal of a CI/CD pipeline is to deliver working software to users quickly and frequently.

- which will increase revenue by releasing value-generating features more quickly

- **Better code quality**

Testing your code's behavior is an essential step in the software release process but doing it thoroughly can also be extremely time consuming.

- which will reduce cost by reducing developer time on issues from new developer code

- **Maximized creativity**

building a CI/CD pipeline eliminates waste and helps create a leaner, more efficient software development and release process.

- which will avoid cost by reducing human error, Faster deployments

- **Reduced risk of defects**

Finding and resolving defects late in the development process is costly and time-consuming. This is particularly true when problems arise with features already released to production.

- which will avoid cost by reducing bugs in production and less time in testing

- **Quick rollback if required**

One of the most exclusive benefits of a CI/CD pipeline is that it leads to the quick and easy rollback of code changes if there are any issues in the production environment after a release.

- which will protect revenue by Quickly returning production to working state

THE CHALLENGES OF CI/CD

- **Performance Issues**

If CI/CD implementation is not done correctly, performance issues such as slow page loading, sluggish server responses, and memory optimization can affect your software speed, responsiveness, stability, and overall scalability.

- **Team Communication**

When you are working within a team, if something fails during the deployment, then you need to communicate with your team to resolve it quickly. Moreover, problems may arise if an automated build test finds an error and does not communicate it.

- **Flawed Automated Testing**

Flawed automated testing systems give nightmares to developers. Many checks are performed in the CI/CD pipeline to ensure high-quality builds and that the code is deployed quickly. But what if the pipeline has a flawed testing mechanism? This will lead to deploying faulty codes and disrupting the performance.

BEST PRACTICES FOR CI/CD

- **Fail Fast**
- Set up your CI/CD pipeline to find and reveal failures as fast as possible. The faster you can bring your code failures to light, the faster you can fix them.
- **Measure Quality**
- Measure your code quality so that you can see the positive effects of your improvement work (or the negative effects of technical debt).
- **Only Road to Production**
- Once CI/CD is deploying to production on your behalf, it must be the only way to deploy. Any other person or process that meddles with production after CI/CD is running will inevitably cause CI/CD to become inconsistent and fail.
- **Maximum Automation**
- If it can be automated, automate it. This will only improve your process!
- **Config in Code**
- All configuration code must be in code and versioned alongside your production code. This includes the CI/CD configuration files!

CONCLUSION

- **Developing a CI/CD pipeline** is a standard practice for businesses that frequently improve applications and require a reliable delivery process. Once in place, the CI/CD pipeline lets the team focus more on enhancing applications and less on the details of delivering it to various environments.
- **Getting started with CI/CD** requires devops teams to collaborate on technologies, practices, and priorities. Teams need to develop consensus on the right approach for their business and technologies. Once a pipeline is in place, the team should follow CI/CD practices consistently.