CI/CD BENEFITS PROPOSAL

BY: AHMED SAEED

Content

- What is CI/CD ?
- What are our current pain points ?
- How continuous integration improves collaboration and code quality ?
- What are the challenges we will be confronted with ?

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.

- CI/CD is a best practice for devops teams. It is also a best practice in agile methodology. By automating integration and delivery, CI/CD lets software development teams focus on meeting business requirements while ensuring code quality and software security.

- Continuous integration is a coding philosophy and set of practices that drive development teams to frequently implement small code changes and check them in to a version control repository. Most modern applications require developing code using a variety of platforms and tools, so teams need a consistent mechanism to integrate and validate changes. Continuous integration establishes an automated way to build, package, and test their applications. Having a consistent integration process encourages developers to commit code changes more frequently, which leads to better collaboration and code quality.

What are our current pain points

- Our manual release process is error-prone and always leads to delays of production deployments.
- This in turn often leads to poor software quality since we don’t have time for quality analysis anymore.
- Deployments are complex. Only a chosen few experts are able to understand the whole process and tons of hand-crafted helper scripts. No smoke tests and rollback mechanisms.
- We get late feedback from the business department which prevents us from creating flexible solutions.

How continuous integration improves collaboration and code quality

- Continuous integration is a development philosophy backed by process mechanics and automation. When practicing continuous integration, developers commit their code into the version control repository frequently; most teams have a standard of committing code at least daily. The rationale is that it's easier to identify defects and other software quality issues on smaller code differentials than on larger ones developed over an extensive period. In addition, when developers work on shorter commit cycles, it is less likely that multiple developers will edit the same code and require a merge when committing.

- Teams implementing continuous integration often start with the version control configuration and practice definitions. Although checking in code is done frequently, agile teams develop features and fixes on shorter and longer timeframes. Development teams practicing continuous integration use different techniques to control what features and code are ready for production.

How continuous integration improves collaboration and code quality - Cont'd

- Many teams use feature flags, a configuration mechanism to turn features and code on or off at runtime. Features that are still under development are wrapped with feature flags in the code, deployed with the main branch to production, and turned off until they are ready to be used. In recent research, devops teams using feature flags had a ninefold increase in development frequency. Feature flagging tools such as CloudBees, Optimizely Rollouts, and LaunchDarkly integrate with CI/CD tools to support feature-level configurations.

What are the challenges we will be confronted with

- 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.
- 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.
- Establishing CI/CD comes with a high amount of initial cost and learning. At first sight this might seem overwhelming compared to current best practices
- Version control is necessary because all your components and processes work on a stable version. However, if any process is updated unexpectedly, the entire CI/CD pipeline is broken because of compatibility issues. The biggest problem you can face is an automated system update which will force some processes to update to newer versions, ruining the whole CI/CD cycle.
- Some security vulnerabilities in the CI/CD pipeline make it prone to cyberattacks. Any sensitive information in the pipeline can be stolen by the attacker, which can be disastrous.