Dev-ops assignment-1

Name: CHANDRASHEKAR B
USN: 4N19IS025
Section: A

Topic – Continuous integration and continuous deployment (CI/CD)

1. Continuous Integration and continuous deployment

Continuous integration

Developers practicing continuous integration merge their changes back to the main branch as often as possible. The developer's changes are validated by creating a build and running automated tests against the build. By doing so, you avoid integration challenges that can happen when waiting for release day to merge changes into the release branch.

Continuous integration puts a great emphasis on testing automation to check that the application is not broken whenever new commits are integrated into the main branch.

Continuous delivery

Continuous delivery is an extension of continuous integration since it automatically deploys all code changes to a testing and/or production environment after the build stage.
This means that on top of automated testing, you have an automated release process and you can deploy your application any time by clicking a button.

In theory, with continuous delivery, you can decide to release daily, weekly, fortnightly, or whatever suits your business requirements. However, if you truly want to get the benefits of continuous delivery, you should deploy to production as early as possible to make sure that you release small batches that are easy to troubleshoot in case of a problem.

What is a continuous delivery pipeline?

A continuous delivery pipeline is a series of automated processes for delivering new software. It's an implementation of the continuous paradigm, where automated builds, tests, and deployments are orchestrated as one release workflow. Put more plainly, a CD pipeline is a set of steps your code changes go through to make their way to production.

A CD pipeline delivers, as per business needs, quality products frequently and predictably from test to staging to production in an automated fashion.

For starters, let’s focus on the three concepts: quality, frequency, and predictability.

We emphasize quality to underscore that it’s not traded for speed. Business doesn’t want us to build a pipeline that can shoot faulty code to production at high speed. We will go through the principles of “Shift Left” and “DevSecOps”, and discuss how we can move quality and security upstream in the software development life cycle (SDLC). This will put to rest any concerns regarding continuous delivery pipelines posing risks to businesses.
Frequency indicates that pipelines execute at any time to release features since they are programmed to trigger with commits to the codebase. Once the pipeline MVP (minimum viable product) is in place, it can execute as many times as it needs to with periodic maintenance costs. This automated approach scales without burning out the team. This also allows teams to make small incremental improvements to their products without the fear of a major catastrophe in production.

What is the DevOps pipeline?

Cliche as it may sound, the nation of “to err is human” still holds true. Teams brace for impact during manual releases since those processes are brittle. Predictability implies that releases are deterministic in nature when done via continuous delivery pipelines. Since pipelines are programmable infrastructure, teams can expect the desired behavior every time. Accidents can happen, of course, since no software is bug-free. However, pipelines are exponentially better than manual error-prone release processes, since, unlike humans, pipelines don’t falter under aggressive deadlines.

Pipelines have software gates that automatically promote or reject versioned artifacts from passing through. If the release protocol is not honored, software gates remain closed, and the pipeline aborts. Alerts are generated and notifications are sent to a distribution list comprising team members who could have potentially broken the pipeline.

And that’s how a CD pipeline works: A commit, or a small incremental batch of commits, makes its way to production every time the pipeline runs successfully. Eventually, teams ship features and ultimately products in a secure and auditable way.

Why is CI/CD important?

CI/CD allows organizations to ship software quickly and efficiently. CI/CD facilitates an effective process for getting products to market faster than ever before, continuously delivering code into production, and ensuring an ongoing flow of new features and bug fixes via the most efficient delivery method.

What is the difference between CI and CD?

Continuous integration (CI) is practice that involves developers making small changes and checks to their code. Due to the scale of requirements and the number of steps involved, this process is automated to ensure that teams can build, test, and package their applications in a reliable and repeatable way. CI helps streamline code changes, thereby increasing time for developers to make changes and contribute to improved software.

Continuous delivery (CD) is the automated delivery of completed code to environments like testing and development. CD provides an automated and consistent way for code to be delivered to these environments.

Continuous deployment is the next step of continuous delivery. Every change that passes the automated tests is automatically placed in production, resulting in many production deployments.

Continuous deployment should be the goal of most companies that are not constrained by regulatory or other requirements.

In short, CI is a set of practices performed *as developers are writing* code, and CD is a set of practices performed *after* the code is completed.

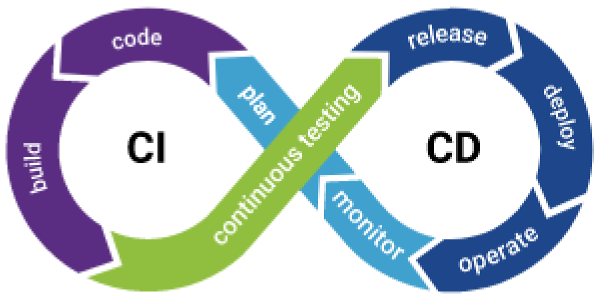

Fig1.1 CI/Cd Flow Diagram

How does CI/CD relate to DevOps?

DevOps is a set of practices and tools designed to increase an organization’s ability to deliver applications and services faster than traditional software development processes. The increased speed of DevOps helps an organization serve its customers more successfully and be more competitive in the market. In a DevOps environment, successful organizations “bake security in” to all phases of the development life cycle, a practice called *DevSecOps*.

The key practice of DevSecOps is integrating security into all DevOps workflows. By conducting security activities early and consistently throughout the software development life cycle (SDLC), organizations can ensure that they catch vulnerabilities as early as possible, and are better able to make informed decisions about risk and mitigation. In more traditional security practices, security is not addressed until the production stage, which is no longer compatible with the faster and more agile DevOps approach. Today, security tools must fit seamlessly into the developer workflow and the CI/CD pipeline in order to keep pace with DevOps and not slow development velocity.

The CI/CD pipeline is part of the broader DevOps/DevSecOps framework. In order to successfully implement and run a CI/CD pipeline, organizations need tools to prevent points of friction that slow down integration and delivery. Teams require an integrated toolchain of technologies to facilitate collaborative and unimpeded development efforts.

What AppSec tools are required for CI/CD pipelines?

One of the largest challenges faced by development teams using a CI/CD pipeline is adequately addressing security. It is critical that teams build in security without slowing down their integration and delivery cycles. Moving security testing to earlier in the life cycle is one of the most important steps to achieving this goal. This is especially true for DevSecOps organizations that rely on automated security testing to keep up with the speed of delivery.

Implementing the right tools at the right time reduces overall DevSecOps friction, increases release velocity, and improves quality and efficiency.

What are the benefits of CI/CD?

- Automated testing enables continuous delivery, which ensures software quality and security and increases the profitability of code in production.
- CI/CD pipelines enable a much shorter time to market for new product features, creating happier customers and lowering strain on development.
- The great increase in overall speed of delivery enabled by CI/CD pipelines improves an organization’s competitive edge.
- Automation frees team members to focus on what they do best, yielding the best end products.
- Organizations with a successful CI/CD pipeline can attract great talent. By moving away from traditional waterfall methods, engineers and developers are no longer bogged down with repetitive activities that are often highly dependent on the completion of other tasks.

Benefits of the CI/CD pipeline

Automation of software releases — from initial testing to the final deployment — is a significant benefit of the CI/CD pipeline. Additional benefits of the CI/CD process for development teams include the following:

- **Reducing time to deployment through automation:** Automated testing makes the development process more efficient, reducing the length of the software delivery process. In addition, continuous deployment and automated provisioning allow a developer's changes to a cloud application to go live within minutes of writing them.
- **Decreasing the costs associated with traditional software development:** Fast development, testing and production (facilitated by automation) means less time spent in development and, therefore, less cost.
- **Continuous feedback for improvement:** The CI/CD pipeline is a continuous cycle of build, test and deploy. Every time code is tested, developers can quickly take action on the feedback and improve the code.
- **Improving the ability to address error detection earlier in the development process:** In continuous integration, testing is automated for each version of code built to look for issues integration. These issues are easier to fix the earlier in the pipeline that they occur.
- **Improving team collaboration and system integration**. Everyone on the team can change code, respond to feedback and quickly respond to any issues that occur