<u>CI/CD</u>

➢ **What are the fundamental differences between DevOps & Agile?**

- DevOps is a practice of bringing development and operations teams together whereas Agile is an iterative approach that focuses on collaboration, customer feedback and small rapid releases.
- DevOps focuses on constant testing and delivery while the Agile process focuses on constant changes.
- DevOps requires relatively a large team while Agile requires a small team.
- DevOps leverages both shifts left and right principles, on the other hand, Agile leverage shift-left principle.
- The target area of Agile is Software development whereas the Target area of DevOps is to give end-to-end business solutions and fast delivery.
- DevOps focuses more on operational and business readiness whereas Agile focuses on functional and non-function readiness.

➢ **What is the need for DevOps?**

- Shorter Development Cycles, Faster Innovation
- Reduce Implementation Failure, Reflections and Recovery Time
- Better Communication and Cooperation
- Greater Competencies
- Reduce Costs and IT Staff

➢ **What are the advantages of DevOps?**

The advantage of DevOps

By creating a more responsive development environment that is closely aligned to business requirements and which removes human error from the project lifecycle, DevOps enables organizations to:

- Reduce the implementation time of new services from months to minutes
- Increase productivity of business and IT teams
- Save costs on maintenance and upgrades, and eliminate unnecessary capital expenditure
- Standardize processes for easy replication and faster delivery
- Improve quality, reliability and reusability of all system components
- Increase the rate of success for digitalization strategies and transformation projects

- Ensure that money invested in cloud infrastructure, analytics and data management are not wasted

- **Explain with a use case where DevOps can be used in industry/ real-life.**

Application of DevOps in the Online Financial Trading Company
The methodology in the process of testing, building, and development was automated in the financial trading company. Using the DevOps, deployment was being done within 45 seconds. These deployments used to take long nights and weekends for the employees. The time of the overall process reduced and the interest of clients increased.

- **What are the success factors for Continuous Integration?**

Implementing the tools for Continuous Integration is the easy part. Making best use of Continuous Integration is the complex bit. Are you making the best use of your continuous integration setup? Here are the things you would need to consider.

- How often is code committed? If code is committed once a day or week, the CI setup is under utilised. Defeats the purpose of CI.
- How is a failure treated? Is immediate action taken? Does failures promote fun in the team?
- What steps are in continuous integration? More steps in continuous integration means more stability.
  - Compilation
  - Unit Tests
  - Code Quality Gates
  - Integration Tests
  - Deployment
  - Chain Tests
- More steps in continuous integration might make it take more time but results in more stable application. A trade-off needs to be made.
  - Run Steps a,b,c on a commit.
  - Run Steps d & e once every 3 hours.
- How long does a Continuous Integration build run for?
  - One option to reduce time taken and ensure we have immediate feedback is to split the long running tests into a separate build which runs less often.

- **What are the differences between continuous integration, continuous delivery, and continuous deployment?**

The Continuous Integration (CI), Continuous Delivery (CD), and Continuous Deployment (CD) process is a framework that enables this approach. The biggest difference between these stages (CI/CDs) is whom it benefits most at each stage.

CI benefits developers most because it allows for code produced to be automatically tested and continuously “integrated” with other developers’ code, and with the existing codebase.

CI/CD

The developer benefits from receiving continuous and immediate feedback regarding code and integration errors. As s/he fixes these errors, automated testing tools in this stage will report if the errors were successfully fixed and when the code is accepted. This continuous feedback loop dramatically increases a developer's productivity.

Continuous Delivery benefits business users because as soon as code is successfully accepted in the CI stage and a logical function can be tested, it is released to users. They verify that the features meet their expectations and provide feedback to developers who then address the feedback in this stage. This feedback loop between users and developers is continuous and seamless. Whereas in traditional waterfall method, users could wait weeks/months to see the features for the first time, this approach can dramatically reduce the time to just hours/days.

Continuous Deployment seamlessly pushes code that has been successfully accepted in the CI/CD cycle into the production environment. This stage benefits all key stakeholders, from application investors who fund the development to external consumers and internal end-users as new features/application is available for immediate (external) commercial sale or internal use.

- **What role does the Quality Assurance (QA) team play in DevOps?**

QA ties together development and operations and enables them to collaborate to have software and applications up & running. Everyone in the organization takes responsibility for quality and stability, and thereby for the business success.

- **Describe an efficient workflow for continuous integration**

Pipeline Stages

- A typical continuous delivery pipeline manages code changes as they move from a version control system to the production server. Let's outline the key touchpoints along the delivery pipeline:

- **Version Control.**
  Many teams manage their code bases through version control software based on Git, which is an open source distributed version control system. There are a number of code management tools out there based on Git workflows, like GitHub, Bitbucket, and GitLab.
  There are two features that have made Git the dominant version control system: the ability to clone, or pull down, a local working copy of a repository, and ability to create branches within a code base. Branches allow you to isolate and test your changes on

a separate area of your repository, allowing the master branch to represent the record of truth for your code base. When you’re satisfied with changes made to a feature branch, it can be merged back in with the master.
Version control systems also allow you to see a complete history of all of the changes that have been made to a repository. This makes it easy to run diff checks and, if necessary, roll back changes that have caused problems.

- **Staging.**
  A staging environment mirrors the environment that your code will run in when it’s fully live, but it’s isolated from the live environment so you can run tests. In the context of building a website, your staging environment would look like a copy of the live site; if you were building an application, your staging environment would be a place where you can push a test build of your application to your host server.

- **Production.**
  The production environment is the live version of your website and app. Crucially, it’s where your users will actually be interfacing with your code — so it’s important to QA any changes before they reach this point.

➢ **What are the best practices for DevOps implementation?**

The best practices of DevOps are:

- Configuration Management
- Continuous Integration
- Automated Testing
- Infrastructure as Code
- Continuous Delivery
- Continuous Deployment
- Continuous Monitoring

CI/CD

- **How will you approach when a project needs to implement DevOps?**

Align your IT goals with Business goals. The need for implementation of DevOps should be business-driven. It should not be implemented just because it is the latest trend, but your development process for the business goals should demand this change.