## What is DevOps?

## DevOps is a culture which promotes collaboration between Development and Operations Team to deploy code to production faster in an automated & repeatable way. The word 'DevOps' is a combination of two words 'development' and 'operations.'

DevOps helps to increases an organization's speed to deliver applications and services. It allows organizations to serve their customers better and compete more strongly in the market.

It is a newly emerging term in IT field, which is nothing but a practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals. It focuses on delivering software product faster and lowering the failure rate of releases.

**2. SDLC:**

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software’s. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

Phase 1: Requirement collection and

Phase 2: analysis

Phase 3: Design:

Phase 4: Coding:

Phase 5: Testing:

Phase 6: Installation/Deployment:

Phase 7: Maintenance:

**3. SDLC MODELS:**

Waterfall Model

Agile Methodologies

**4. WATERFALL MODEL**

Waterfall Model is a sequential model that divides software development into different phases. Each phase is designed for performing specific activity during SDLC phase. It was introduced in 1970 by Winston Royce.

Waterfall model can be used when

* Requirements are not changing frequently
* Application is not complicated and big
* Project is short
* Requirement is clear
* Environment is stable
* Technology and tools used are not dynamic and is stable
* Resources are available and trained

**5. AGILE METHODLOGY**

AGILE methodology is a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project. Both development and testing activities are concurrent unlike the Waterfall model

he agile software development emphasizes on four core values.

1. Individual and team interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

**BLUE/GREEN DEPLOYMENT**

Blue/Green Deployment pattern addresses one of the most important challenges faced by automatic deployments; i.e., the cutover from final stage of testing to live production. Ensuring this cutover happens quickly is the best way to reduce system downtime.

In Blue/Green Deployment approach, the team first ensures two identical production environments; but, only one among them is LIVE at any given point of time.

The LIVE environment is considered as Blue environment, and as the team prepares the next release of their software, they conduct their final stage of testing; considering it as Green environment. Once the team completes their final round of testing, they utilize some sort of tools which enable the user requests directly to the new Green environment. The Blue environment is considered as idle. The basic idea here is to easily switch between the environments.

**Advantages:**

1. A stand-by production node is always running.
2. Allows rollback testing with every release.
3. Rollback option is always available.

**Disadvantages:**

1. Cost of maintaining 2 production environments.
2. Databases are a challenge that may require refactoring.

**CANARY DEPLOYMENT**

Canary release is a pattern that reduces the risk of introducing a new version of software into production; this is done by making it available in a controlled manner to a subset of users before making it available to the entire user set.

In this pattern, the team deploys the new version only to a subset of the infrastructure, where there are no user request routed. Once the team is happy with the performance tests, they start routing a selected group of users to the application. As and when they feel confident, the company can release more servers as the team also routes more user requests.

**Advantages:**

1. Slow Ramp-up/down helps in monitoring/metrics.
2. Ability to do capacity testing.
3. Very safe rollback strategy.

**Disadvantages:**

1. Managing more than one software at a time.
2. Hard to work on distributed software.

**Q1. What is Jenkins?**

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

* First, a developer commits the code to the source code repository. Meanwhile, the Jenkins server checks the repository at regular intervals for changes.
* Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build.
* If the build fails, then the concerned team will be notified.
* If built is successful, then Jenkins deploys the built in the test server.
* After testing, Jenkins generates a feedback and then notifies the developers about the build and test results.
* It will continue to check the source code repository for changes made in the source code and the whole process keeps on repeating.

GIT