GIT

1. What is Git? What is its importance in SDLC?

* Git is a version control system for tracking changes in files and coordinating work on those files among multiple people. It is primarily used for source code management in software development. It is a distributed revision control system and is very useful to support software development workflows.
* So why did Git become so widely adopted, and how did it grow into the first choice of today’s developers? Mostly because it was an enormous step forward for version control.
  + Git is distributed
  + Git defines branching and merging
  + Git has a staging area

1. Difference between GIT and SVN.

GIT:

* Git is open source distributed vice control system developed by Linus Torvalds in 2005. It emphasis on speed and data integrity
* Git has a Distributed Model
* In git every user has their own copy of code on their local like their own branch.
* In git we do not required any Network to perform git operation
* Git is more difficult to learn. It has more concepts and commands.
* Git deals with large number of files like binary files that change quickly that why it become slow.
* In git we create only .git directory
* It does not not have good UI as compared to SVN

SVN:

* Apache Subversion is an open source software version and revision control system under Apache license.
* SVN has a Centralized Model.
* In SVN there is central repository has working copy that also make changes and committed in central repository
* In SVN we required Network for runs the SVN operation
* SVN is much easier to learn as compared to git
* SVN control the large number of binary files easily.
* In SVN we create .svn directory in each folder.
* SVN has simple and better user interface

1. WHAT ARE THE ADVANTAGES OF USING GIT?

* Distributed model: This means your work is your own. You can let others see only what is necessary. Not everything has to be public. There are other advantages to the distributed model, such as the speed (since most everything is local) and possibility of working offline
* Branching and merging are easy: Branching is a walk in the park. It feels like a natural part of the workflow. They are cheap (fast and consume very little space) so that you can branch whenever you want. This means you can sandbox your features and ideas till they are ready for the mainstream.
* Workflow is flexible: Compared to Centralized VCS, git has the qualities that allow to choose your own workflow. It can be as simple as a centralised workflow to as hierarchical as the dictator-lieutenant workflow. Use the process that best fits you.
* Data integrity is assured: Because git uses SHA1 trees, data corruption due to external reasons can be easily detected.

**Icing on the cake:**

* Fast: Git is very fast, even when compared to other DVCS, for local as well as network operations
* Staging area: Make sure your commits have logically grouped changes and not everything else you are working on
* Free: I am sure you dont want to spend 450$ for your personal project. Your manager will appreciate it if you save him N x 450$

1. WHAT IS ‘STAGING AREA’ OR ‘INDEX’ IN GIT?

* The **index**, or staging area, is where commits are prepared. The index compares the files in the working tree to the files in the repo. When you make a change in the working tree, the index marks the file as modified before it is committed

1. WHAT IS GIT STASH?

* In Git, the stash operation takes your modified tracked files, stages changes, and saves them on a stack of unfinished changes that you can reapply at any time.

1. WHAT IS THE FUNCTION OF GIT CLONE?

* The git clone command is used to create a copy of a specific repository or branch within a repository.
* Git is a distributed version control system. Maximize the advantages of a full repository on your own machine by cloning.
* When you clone a repository, you don't get one file, like you may in other centralized version control systems. By cloning with Git, you get the entire repository - all files, all branches, and all commits.

7.HOW CAN YOU CREATE A REPOSITORY IN GIT?

* To create a new repo, you'll use the git init command. git init is a one-time command you use during the initial setup of a new repo. Executing this command will create a new .git subdirectory in your current working directory. This will also create a new master branch.

8.WHAT IS THE PURPOSE OF BRANCHING IN GIT

* In Git, branches are a part of your everyday development process. Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes. This makes it harder for unstable code to get merged into the main code base, and it gives you the chance to clean up your future's history before merging it into the main branch.

9.WHAT IS THE DIFFERENCE BETWEEN ‘git remote’ AND ‘git clone’

* git remote add just creates an entry in your git config that specifies a name for a particular URL. You must have an existing git repo to use this.
* git clone creates a new git repository by copying an existing one located at the URI you specify
* **git clone:**
  + Will physically download the files into your computer. It will take space from your computer. If the repo is 200Mb, then it will download that all and place it in the directory you cloned
* **git remote add:**
  + Won't take space! It's more like a pointer! It doesn't increase your disk consumption. It just gets a snapshot of what branches are available and their git commit history I believe. It doesn't contain the actual file/folders of your project.

10. WHAT IS THE FUNCTION OF git diff IN GIT

* Diffing is a function that takes two input data sets and outputs the changes between them. git diff is a multi-use Git command that when executed runs a diff function on Git data sources. These data sources can be commits, branches, files and more.

11. EXPLAIN WHAT COMMIT MESSAGE IS?

* The **commit** command is used to save changes to a local repository after staging in Git. However, before you can save changes in Git, you have to tell Git which changes you want to save as you might have made tons of edits. A great way to do that is by adding a **commit message** to identify your changes.
* Commit messages can adequately communicate why a change was made, and understanding that makes development and collaboration more efficient.

12.WHY IS IT ADVISABLE TO CREATE AN ADDITIONAL COMMIT RATHER THAN AMENDING AN EXISTING COMMIT?

* Git amend internally creates a new commit and replaces the old commit. If commits have already been pushed to central repository, it should not be used to modify the previous commits.
* It should be generally used for only amending the git comment.

13. WHAT IS REBASING?

* Rebasing is the process of moving or combining a sequence of commits to a new base commit. Rebasing is most useful and easily visualized in the context of a feature branching workflow.

**MAVEN FUNDAMENTALS**

1. Explain what is Maven? How does it work?
   * Maven is a popular open source build tool for enterprise Java projects, designed to take much of the hard work out of the build process. Maven uses a declarative approach, where the project structure and contents are described, rather then the task-based approach used in Ant or in traditional make files, for example. This helps enforce company-wide development standards and reduces the time needed to write and maintain build scripts.
   * **Core Concepts of Maven:**
     + **POM Files:**Project Object Model(POM) Files are XML file that contains information related to the project and configuration information such as dependencies, source directory, plugin, goals etc. used by Maven to build the project. When you should execute a maven command you give maven a POM file to execute the commands. Maven reads pom.xml file to accomplish its configuration and operations.
     + **Dependencies and Repositories:**Dependencies are external Java libraries required for Project and repositories are directories of packaged JAR files. The local repository is just a directory on your machine hard drive. If the dependencies are not found in the local Maven repository, Maven downloads them from a central Maven repository and puts them in your local repository.
     + **Build Life Cycles, Phases and Goals:**A build life cycle consists of a sequence of build phases, and each build phase consists of a sequence of goals. Maven command is the name of a build lifecycle, phase or goal. If a lifecycle is requested executed by giving maven command, all build phases in that life cycle are executed also. If a build phase is requested executed, all build phases before it in the defined sequence are executed too.
     + **Build Profiles:**Build profiles a set of configuration values which allows you to build your project using different configurations. For example, you may need to build your project for your local computer, for development and test. To enable different builds you can add different build profiles to your POM files using its profiles elements and are triggered in the variety of ways
     + **Build Plugins:**Build plugins are used to perform specific goal. you can add a plugin to the POM file. Maven has some standard plugins you can use, and you can also implement your own in Java
2. Explain what is POM? What is significance?
   * A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects. Examples for this is the build directory, which is target; the source directory, which is src/main/java; the test source directory, which is src/test/java; and so on. When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.
   * Some of the configuration that can be specified in the POM are the project dependencies, the plugins or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.
3. Explain what the maven artefact is?

* An artifact is a file, usually a JAR, that gets deployed to a Maven repository.
* A Maven build produces one or more artifacts, such as a compiled JAR and a "sources" JAR.
* Each artifact has a group ID (usually a reversed domain name, like com.example.foo), an artifact ID (just a name), and a version string. The three together uniquely identify the artifact.
* A project's dependencies are specified as artifacts.

1. LIST OUT THE DEPENDECY SCOPE IN MAVEN.

* Dependency scope is used to limit the transitivity of a dependency and to determine when a dependency is included in a classpath.
* There are 6 scopes:
  + compile
  + provided
  + runtime
  + test
  + system
  + import

1. LIST OUT WHAT ARE THE BUILD PHASES IN MAVEN.

* validate
* compile
* test-compile
* test
* package
* integration-test
* install
* deploy

1. MENTION THE 3 BUILD LIFECYCLE OF MAVEN.

* default: the main life cycle as it's responsible for project deployment
* clean: to clean the project and remove all files generated by the previous build
* site: to create the project's site documentation

1. LIST OUT WHAT ARE THE ASPECTS DOES MAVEN MANAGES.

Maven handles following aspects,

* Build.
* Documentation.
* Reporting.
* Dependencies.
* SCMs.
* Releases.
* Distribution.
* Mailing list.

1. EXPLAIN WHAT A MAVEN REPOSITORY IS? WHAT ARE THEIR TYPES?

* A **maven repository** is a directory of packaged JAR file with pom.xml file. Maven searches for dependencies in the repositories. There are 3 types of maven repository:

1. Local Repository
2. Central Repository
3. Remote Repository
   * + Maven searches for the dependencies in the following order:

* **Local repository** then **Central repository** then **Remote repository**
  + - If dependency is not found in these repositories, maven stops processing and throws an error.

1. EXPLAIN HOW YOU CAN EXCLUDE DEPENDENCY?
   * + Since Maven resolves dependencies transitively, it is possible for unwanted dependencies to be included in your project's classpath. For example, a certain older jar may have security issues or be incompatible with the Java version you're using. To address this, Maven allows you to exclude specific dependencies. Exclusions are set on a specific dependency in your POM, and are targeted at a specific groupId and artifactId. When you build your project, that artifact will not be added to your project's classpath *by way of the dependency in which the exclusion was declared*.
     + How to use dependency exclusions?
       - Add an <exclusions> element in the <dependency> element by which the problematic jar is included.
2. For POM, what are the minimum required elements?

* The minimum requirement for a POM are the following:
* project root
* modelVersion - should be set to 4.0.0
* groupId - the id of the project's group.
* artifactId - the id of the artifact (project)
* version - the version of the artifact under the specified group

CI/CD

1.WHAT ARE THE FUNDAMENTAL DIFFERENCES BETWEEN DevOps AND 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.

2. WHAT IS THE NEED FOR DEVOPS?

* DevOps is a methodology just like agile or waterfall, But it differs in the matter that It improves the collaboration between developers and the testing team or in better terms DevOps is a culture. It helps in development & deployment of software but in a much more efficient and faster manner. It does so by automating a lot of the steps that would be previously left upon humans to work.
* If we were to consider software Development was divided into 4 parts:
  + Development => Testing => Deployment => Monitoring
* These process can be easily automated using the DevOps tools and hence decrease the time it takes to release a software or an update of that software.
* Why we need DevOps?
  + A Scenario: Now imagine yourself working in a company with a Development team and a Testing team, if you assume that the Developer works in a Windows environment and the tester works in a Mac OS environment then when the Dev sends his code to the tester then there is a chance that the same code wouldn’t work in the tester’s system as they both use different environments.
  + To solve a problem like the one above the idea of DevOps comes in very handy, in the above scenario this problem wouldn’t happen if there was a DevOps tool like Docker used to enforce a standard environment for all the people working on a particular piece of code.
  + More reasons to use DevOps:
    - It makes the software development process much more productive
    - smaller and more frequent updates are possible
    - Teams are divided to concentrate on specific tasks
    - Easy rollback of software is possible
    - frequent releases mean frequent feedback and hence development of a better product is possible
    - It also reduces the tension between teams as there are less clashes and misunderstandings creating a friendly work environment
    - reduces the cost to the company over a period of time

1. WHAT ARE THE ADVANTAGES OF DEVOPS?

#### Technical Benfits:

* Continuous software delivery
* Less complexity to manage
* Faster resolution of problems

#### Cultural Benefits

* Happier, more productive teams
* Higher employee engagement
* Greater professional development opportunities

#### Business Benefits

* Faster delivery of features
* More stable operating environments
* Improved communication and collaboration
* More time to innovate (rather than fix/maintain)

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

### ****Deliver value to Customers****

* DevOps ***minimizes the time*** it takes to deliver value to customers. The cycle time from the developer’s completion of a story/task until production reduces significantly, allowing the value to be realized as quickly as possible.
* The most important value realized through DevOps is that it allows IT organizations to ***focus on their “core” business activities***.  By removing constraints within the value stream and automating deployment pipelines, teams can focus on the activities. This helps in creating customer value rather than just moving bits and bytes. These activities increase the sustainable competitive advantage of a company and create better business outcomes.

### ****Reduced cycle time****

* Internally DevOps is the only way to achieve the agility to deliver secure code with insights. It is important to have gates and a well-crafted DevOps process. When you are delivering a new version, it can run side-by-side with the current version. You can also compare metrics to accomplish what you wanted to with application and performance metrics.
* DevOps drive development teams towards **continuous improvement and faster release cycles**. If done well, this iterative process allows more focus over time, on the things that really matter. Such as things that create great experiences for users – and less time on managing tools, processes, and tech

### ****Time to market****

* The most important problem being solved is the **reduction of the complexity of the process.**This contributes significantly towards our business success by shortening our time to market, giving us quick feedback on features, and making us more responsive to our customers’ needs.

### ****Problem Resolution****

* The greatest value of successful DevOps implementation is higher confidence in delivery, visibility, and traceability to what’s going on, so you can solve problems quicker.
* Another important advantage of DevOps is not wasting any time. Aligning an organization’s people and resources enables rapid deployments and updates. This allows DevOps programs to fix problems before they turn into disasters. DevOps creates a culture of transparency that promotes focus and collaboration among development, operations, and security teams.

1. What are the success factors for continuous integration?

* 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.

1. What are the differences between continous integration, continuous delivery 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](https://www.atlassian.com/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.

### Continuous deployment

* + [Continuous deployment](https://www.atlassian.com/continuous-delivery/continuous-deployment) goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production.
  + Continuous deployment is an excellent way to accelerate the feedback loop with your customers and take pressure off the team as there isn't a Release Day anymore. Developers can focus on building software, and they see their work go live minutes after they've finished working on it.

8.WHAT ROLE DOES THE QUALITY ASSURANCE TEAM PLAY IN DevOps?

* In the DevOps approach, testers and developers play equally important roles. The boundaries of job responsibilities between a developer and tester become blurred 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.
* A developer thinks more like a tester and continually checks their own code for potential errors. Conversely, a tester, whose responsibility is to test the application/software using different types of testing tools & methodologies, also fixes the code. In such a scenario, organizations without a strong and definite QA and testing strategy to “continuously test” in DevOps face troubles in meeting their goals for an accelerated speed of delivery.
* Continuous Testing, when implemented diligently, allows flawless continuity in the delivery cycle. It reduces the time from development to operations. QA is moving into a more strategic role of building robust software testing infrastructure while tracking quality across the SDLC.

9.DESCRIBE AN EFFICIENT WORKFLOW FOR CONTINUOUS INTEGRATION?

* A successful workflow when implementing continuous integration encompasses the following practices:
* Implement and maintain a repository for the project’s source code
* Automate the build and integration
* Make the build self-testing
* Commit changes daily to the baseline
* Build all commits added to the baseline
* Keep the builds fast
* Run tests in a clone of the production environment
* Make it easy to get the latest deliverables
* Make build results easy to be monitored by everyone
* Automate deployment

10.WHAT ARE THE BEST PRACTICES FOR DEVOPS IMPLEMENTATION?

### 1. Active Collaboration of Stakeholders

Fundamental to DevOps practices is the close cooperation and active participation of developers, operations staff, support people, and other stakeholders. Initiated by Extreme Programming (XP), “onsite customer” is a common practice aimed at motivating developers to work in close association with the business and to encourage the support staff and operations to cooperate with the developers. Input from all the stakeholders greatly improves the chances of success.

### 2. Automation of Tests and Building Environment

In order to develop quality code, the software needs to be regularly tested by developers. With DevOps, developers get an early testing opportunity to identify and sort out issues. In contrast to manual testing, automated testing enables faster execution of the Software Development Life Cycle and is a significant aspect of code development, middleware configurations, and changes to data and networking.

Automation of the building environment is critical when virtual infrastructure is used; the manual building of environments poses the risk of essential requirements being neglected or reverted to legacy implementations.

### 3. Integrated Configuration and Change Management

Integrated configuration management allows the software development teams to use the existing services in software development and eliminates the hassle of creating new services over and over again. It involves monitoring, automation, maintenance, and management of system configurations taking place across networks, applications, servers, storage, and additional managed services.

Change management appears in the picture whenever the need for a change arises during configuration management. What kind of consequences and opportunities the change may expose and what other systems could be affected at a broader level are essential aspects of change management.

### 4. Continuous Integration and Continuous Deployment

Continuous Integration is a practice referring to the regular up-gradation of the changes in the code in the repository. This tool enables developers to detect challenges in integration and solve them at an early stage, thus improving collaborations among teams and delivering the finest software products.

The process of continuous deployment involves the automatic deployment of the code in the production environment. Several tools are available that carry out continuous deployment right from staging up to production, helping developers to minimize the time between the identification and creation of new features and their deployment in production.

**Also read:** [DevOps Project Ideas](https://www.upgrad.com/blog/devops-projects-for-beginners/)

### 5. Continuous Delivery and Product Support

To minimize the manual work, risk of product failure, and increase the efficiency of deliveries, DevOps adopts the practice of continuous delivery. The developers update the newly developed code; get it assessed by the QA team by both manual and automated methods, and once the case clears all the tests, it is approved for production.

Thus, applications are built, tested, and released in fast, frequent and short cycles. To solve issues relating to the production process, developers resort to the policy of production support to understand and enhance the process of software designing.

### 6. Application Monitoring and Automation of Dashboards

A crucial pre-requisite to optimizing the performance of the application is monitoring the infrastructure of the app that includes operating systems, application servers, and communication services.

Thus, it is vital on the part of the operations and development teams to pro-actively monitor and check the app performance through various tools available for the purpose. Automated Dashboards act as a centralized hub to provide real-time data and insights of each operation, along with an understanding of the right set of tools for automation testing. Thus, the operations teams get a comprehensive view of the changes occurring in the system

11.HOW WILL YOU APPROACH WHEN PROJECT NEEDS TO IMPLEMENT A DEVOPS?

### ****1) Evaluate the need to implement DevOps practice****

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.

### ****2) Break the organizational silos & encourage collaboration****

DevOps practices demand to break down functional silos among various disciplines in IT. The philosophy of DevOps essentially is that development, operations, and other functions must work closely by cooperating and collaborating among themselves.  Breaking down organizational silos improves communication among the teams enabling accessibility to information to everyone about what was done in the past, people involved and the associated results. It helps in better decision making, in turn, fetches better output and better ideas.

### ****3) Put Customer / end-user satisfaction at the center****

Organizations must keep adapting themselves to the ever-changing customer demand and deliver services / solutions that meet, rather exceed, customer expectations regarding time, functionality and performance.  This is possible only by embracing the change in culture that stresses on team effort, transparent communications, and commitment to customer satisfaction, etc. Without the support of all the key business stakeholders, DevOps will not be successful. Right from defining the requirements, prototype development, unit/integration/regression testing, to deployment, everyone should be involved.

### ****4) Don’t jump start, instead, start small and then scale up****

Achieve DevOps approach for faster and smaller release cycles and then adopt at scale. Some quick successes consolidate the belief of various stakeholders in the new approach. Moving the IT culture away from silos need trust and acceptance in the new philosophy. Also, Organizations need to upskill current talent rather than hiring from outside. It enables the existing employees to achieve some early success which helps in improving their confidence on adopting DevOps.

### ****5) Automate wherever possible****

Automation enables faster execution throughout the SDLC, keeping up with the speed of DevOps. Automation can be employed and extended to code development, middleware configuration, database and networking changes, and to essential testing including regression testing and load testing. Automation saves time and efforts of developers, testers and operations personnel and, in turn, total costs.

### ****6) Select tools that are compatible with each other****

The automation tools to be used in DevOps should be chosen depending upon how they react with another tool. It is recommended to choose a toolset which is compatible with your IT environment. Ensure that you adopt tools that are suitable to the rest of the toolchain that is existing. Tooling decisions should be taken wisely considering the overall tool compatibility for your organization. It is usually effective if the tools that you choose are from a single vendor because such tools must have been closely integrated with each other. Careful selection of tools reduce the conflicts that they possibly create between development and operations.

### ****7) Define performance reviews for team and an individual****

When the IT culture has to be collaborative, it requires an evaluation of team’s as well as individual’s performance in the team. Since cooperation and collaboration are at the core of DevOps, performance reviews for developers and operations personnel should mostly be based on their teams’ ability to meeting their development and deployment goals.

### ****8) Ensure real-time visibility into the project****

For a cross-functional IT organization, it is important to have a project management tools that provide real-time visibility into a project or an application is required. It makes the project coordination among different functions easier. All the stakeholders need to understand in which phase the project is exactly in the development to the deployment process. Advanced project management tools have built-in automation that eases getting the information by displaying who and what are the crucial resources for the current tasks of the project.

### ****9) Integrate and deliver continuously****

Embracing DevOps without implementing Continuous integration and Continuous Delivery will be inefficient and unsuccessful. Continuous Integration is one of the key components of agile processes which enables developers to develop a software in small, regular steps by immediately detecting defects and providing feedback.

Continuous delivery is an extension of continuous integration. Continuous Delivery approach ensures that every new or revised requirement is rapidly and safely deployed to production with quality by delivering each and every change to a production-like environment and making sure that the software / application functions as intended through rigorous test automation. It confirms that the software functions as intended through rigorous automated testing. Hence, Continuous Integration and Continuous Delivery should not be neglected for successful DevOps implementation.

### ****10) Achieve better results with monitoring & feedback****

To know if the software or application is performing as desired while the environment is unwavering, continuous monitoring is essential. The Operations team has to ensure that the applications are performing at the optimal levels. They may work with the development team to build monitoring and analytics capabilities right into the applications being developed.

Finally, DevOps is a set of principles and practices that facilitates an organization to make their delivery of software / applications lean and efficient, while leveraging feedback from end-users that help to continuously improve. Feedback mechanism improves the processes of delivering an application.