**GIT:**

1. **What is GIT and its significance 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

One of the biggest advantages of Git is its branching capabilities. Unlike centralized version control systems, Git branches are cheap and easy to merge. This facilitates the feature branch workflow popular with many Git users. Feature branches provide an isolated environment for every change to your codebase.

1. **What is the difference between GIT and SVN?**

**Apache Subversion (SVN):** Apache Subversion is an open-source software version and revision control system under the Apache license. It managed files and folders that are present in the repository. It can operate across the network, which allows it and used by people on different computer .we can say that a repository is like an ordinary file server which allows it to be used by people on a different computer.

**GIT**: Git is an open-source distributed vice control system developed by Linus Torvalds in 2005. Its emphasis on speed and data integrity in which there is no centralized connectivity is needed. It is powerful and cheap branching with easy merge in which each developer has his repository and have a local copy in which they can change history. It supports non-linear development branches and applications with a large number of codes files.

|  |  |
| --- | --- |
| GIT | SVN |
| * 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. * Distributed System. * Branching. * Compatibility. * Non-linear Development. * Lightweight. * Open source. | * 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 * Directories are versioned * Copying, deleting, and renaming. * Free-form versioned metadata . * Atomic commits. * Branching and tagging. * Merge tracking. * File locking. |

1. **What are the advantages of using GIT?**

* Performance: Git provides the best performance when it comes to version control systems. Committing, branching, merging all are optimized for a better performance than other systems.
* Security: Git handles your security with cryptographic method SHA-1. The algorithm manages your versions, files, and directory securely so that your work is not corrupted.
* Branching Model: Git has a different branching model than the other VCS. Git branching model lets you have multiple local branches which are independent of each other. Having this also enables you to have friction-less context switching (switch back and forth to new commit, code and back), role-based code (a branch that always goes to production, another to testing etc) and disposable experimentation (try something out, if does not work, delete it without any loss of code).
* Staging Area: Git has an intermediate stage called “index” or “staging area” where commits can be formatted and modified before completing the commit.
* Distributed: Git is distributed in nature. Distributed means that the repository or the complete code base is mirrored onto the developer’s system so that he can work on it only.
* Open Source: This is a very important feature of any software present today. Being open source invites the developers from all over the world to contribute to the software and make it more and more powerful through features and additional plugins. This has led the Linux kernel to be a software of about 15 million lines of code.

1. **What is “Staging Area” or “Index” in GIT?**

The Git index is used as a staging area between your working directory and your repository. You can use the index to build up a set of changes that you want to commit together. When you create a commit, what is committed is what is currently in the index, not what is in your working directory.

1. **What is GIT stash?**

The git stash command takes your uncommitted changes (both staged and unstaged), saves them away for later use, and then reverts them from your working copy.

1. **What is the function of git clone?**

git clone is primarily used to point to an existing repo and make a clone or copy of that repo at in a new directory, at another location. The original repository can be located on the local filesystem or on remote machine accessible supported protocols. The git clone command copies an existing Git repository.

1. **How can you create a repository in Git?**

* Create a directory to contain the project.
* Go into the new directory.
* Type git init.
* Write some code.
* Type git add to add the files (see the typical use page).
* Type git commit.

1. **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.

1. **What is the difference between ‘git remote’ and ‘git clone’?**

They are two completely different things. git remote is used to refer to a remote repository or your central repository. 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 is used to copy or clone a different repository. git clone creates a new git repository by copying an existing one located at the URI you specify.

1. **What is the function of ‘git diff ’ in git?**

Diff command is used in git to track the difference between the changes made on a file. Since Git is a version control system, tracking changes are something very vital to it. Diff command takes two inputs and reflects the differences between them. It is not necessary that these inputs are files only

1. **Explain what the commit message is?**

git commit -a. Commit a snapshot of all changes in the working directory. This only includes modifications to tracked files (those that have been added with git add at some point in their history). git commit -m "commit message" A shortcut command that immediately creates a commit with a passed commit message.

1. **Why is it advisable to create an additional commit rather than amending an existing commit?**

The amend operation will destroy the state that was previously saved in a commit. If it’s just the commit message being changed then that’s not an issue. But if the contents are being amended then chances of eliminating something important remains more.

1. **What is Rebasing?**

Rebasing a branch updates one branch with another by applying the commits of one branch on top of the commits of another branch. For example, if working on a feature branch that is out of date with a dev branch, rebasing the feature branch onto dev will allow all the new commits from dev to be included in feature.

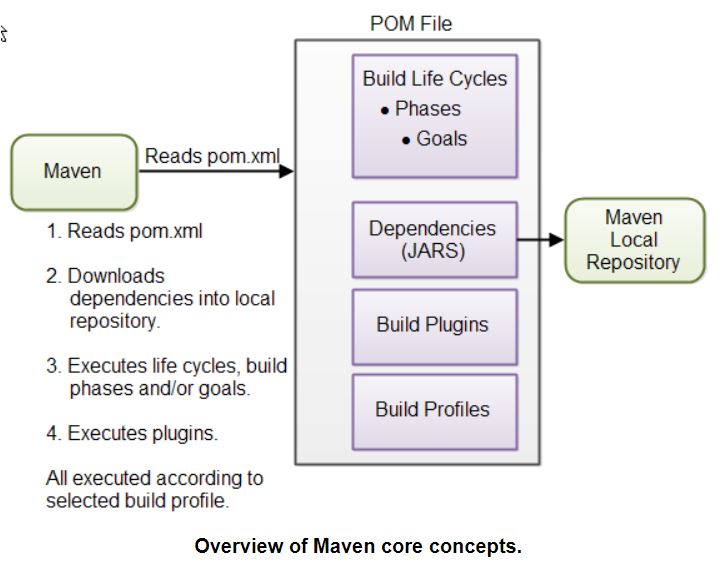
**Maven Fundamentals**

1. **Explain what is Maven? How does it work?**

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation. It simplifies the build process. POM is an XML file that contains information about the project and configuration details used by Maven to build the project. 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.

1. **Explain what is POM and its significance?**

Maven is centered around the concept of POM files (Project Object Model). A POM file is an XML representation of project resources like source code, test code, dependencies (external JARs used) etc. The POM contains references to all of these resources. The POM file should be located in the root directory of the project it belongs to.



1. **Explain what a Maven artifact 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).A project's dependencies are specified as artifacts.

1. **List out the dependency scope in Maven?**

There're two types of dependencies in Maven direct and transitive.Direct dependencies are the ones that are explicitly included in the project. These can be included in the project using <dependency> tags. Transitive dependencies, on the other hand, are dependencies required by our direct dependencies. Required transitive dependencies are automatically included in our project by Maven.



Maven has 6 default dependency scopes.

* + **Compile:**

This is the default scope when no other scope is provided.

* + **Provided:**

****

This scope is used to mark dependencies that should be provided at runtime by JDK or a container,

* + **Runtime:**

The dependencies with this scope are required at runtime, but they're not needed for compilation of the project code.



* + **Test:**

This scope is used to indicate that dependency isn't required at standard runtime of the application, but is used only for test purposes.



* + **System:**

System scope is much similar to the provided scope. The main difference between those two scopes is that system requires us to directly point to specific jar on the system.

* + **Import:**

This scope was added in Maven 2.0.9 and it's only available for the dependency type pom.

1. **List out what are the build phases in Maven?**
   * validate: check if all information necessary for the build is available
   * compile: compile the source code
   * test-compile: compile the test source code
   * test: run unit tests
   * package: package compiled source code into the distributable format (jar ,etc)
   * integration-test: process and deploy the package if needed to run integration tests
   * install: install the package to a local repository
   * deploy: copy the package to the remote repository
2. **Mention the three build lifecycle of Maven?**

There are three built-in build lifecycles: default, clean and site.

The default lifecycle handles your project deployment.The clean lifecycle handles project cleaning, while the site lifecycle handles the creation of your project's site documentation.

1. **List out what are the aspects does Maven Manages?**

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

1. **Explain what a Maven Repository is? What are their types?**

A repository in Maven holds build artifacts and dependencies of varying types.

* The local repository is a directory on the computer where Maven runs. It caches remote downloads and contains temporary build artifacts that you have not yet released.
* Remote repositories refer to any other type of repository, accessed by a variety of protocols such as file:// and https://. These repositories might be a truly remote repository set up by a third party to provide their artifacts for downloading.

1. **Explain how you can exclude dependency?**

* Open the dependency POM and find the transitive dependency you want to exclude. Copy ***groupId*** and ***artifactId.***
* In your project POM, underneath your active dependency, enter ***exclusions*** and using code completion paste the copied info of the dependency you want to exclude.



1. **For POM what are the minimum required elements?**
   * Project root
   * Model Version
   * GroupId
   * ArtifactId
   * Version

**CI/CD**

1. **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.
* 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. DevOps focuses more on operational and business readiness whereas Agile focuses on functional and non-function readiness.

1. **What is the need for DevOps?**

* **Shorter Development Cycles, Faster Innovation**

When development and operations teams are in separate silos, it’s usually difficult to tell if an application is ready for operations. When development teams simply turn over an application, the operations’ cycle times are extended needlessly.With a combined development and operations team, applications are ready for use much more quickly

* **Reduced Deployment Failures, Rollbacks, and Time to Recover**

Part of the reason teams experience deployment failures is due to programming defects. The shorter development cycles with DevOps promote more frequent code releases. This, in turn, makes it easier to spot code defects. Therefore, teams can reduce the number of deployment failures

* **Improved Communication and Collaboration**

DevOps improves the software development culture. Combined teams are happier and more productive. The culture becomes focused on performance rather than individual goals

* **Increased Efficiencies**

Increased efficiency helps to speed the development process and make it less prone to error. There are ways to automate DevOps tasks. Continuous integration servers automate the process of testing code, reducing the amount of manual work required.

* **Reduced Costs**

All of the DevOps benefits translate to reduced overall costs

1. **What are the advantages of DevOps?**

* Quicker mitigation of software defects

With better communication and collaboration between operations and software development, you can identify and mitigate defects at any stage of the development cycle. The same culture can be applied to Application development, where defects prove costlier.

* Better resource management

During the application and software development stage, developers and testers are constantly waiting for resources to arrive causing delays in delivery. Agile with DevOps ensures that the app development arrives in testing phase much quicker than existing operations.

* Reduced human errors

DevOps reduces the chances of human errors during development and operations process by deploying frequent iterations. Lower the application failure rate with multiple deployments in the process in a defined timeline.

* Enhanced version control

Emphasizing on the individuals and interactions, DevOps allows the developers to leverage on programmable dynamic infrastructure at all stages of software/ application development cycle. It allows version control and automated coding options.

* Stable operating environment

Stability is the key to any business platform, and DevOps is established to bring stability with reliability. Organizations with DevOps get their deployment 30 times faster than their rivals with 50% lesser chances of failure.

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

Etsy is a peer-to-peer e-commerce website focused on handmade or vintage items and supplies, as well as unique factory-manufactured items. Etsy struggled with slow, painful site updates that frequently caused the site to go down. It affected sales for millions of Etsy’s users who sold goods through an online marketplace and risked driving them to the competitor.With the help of a new technical management team, Etsy transitioned from its waterfall model, which produced four-hour full-site deployments twice weekly, to a more agile approach. Today, it has a fully automated deployment pipeline, and its continuous delivery practices have reportedly resulted in more than 50 deployments a day with fewer disruptions.

1. **What are the success factors for Continuous Integration?**

Continuous Integration can be defined as Building software and taking it through as many tests as possible with every change.

Different tools for supporting Continuous Integration are Hudson, Jenkins and Bamboo. Jenkins is the most popular one currently. They provide integration with various version control systems and build tools.

1. **What are the differences between continuous 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 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.

* **Continuous deployment**

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.

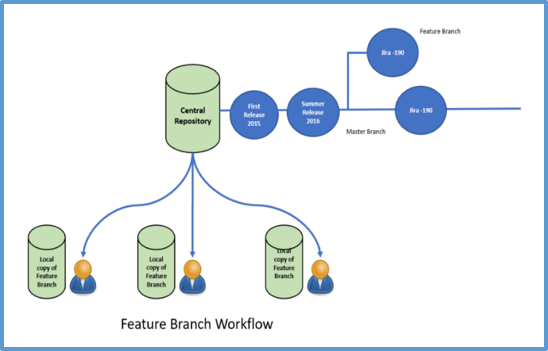
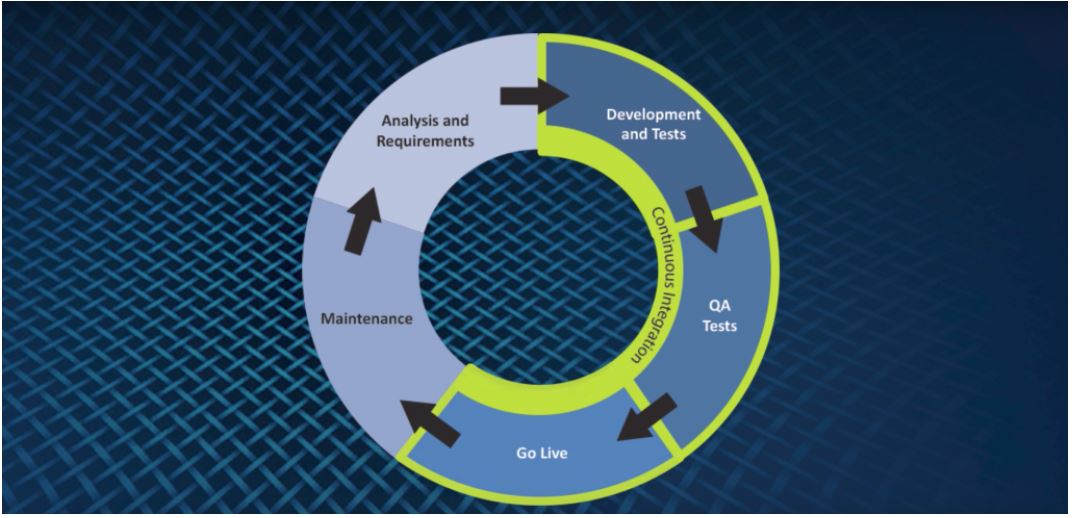
Continuous deployment is an excellent way to accelerate the feedback loop with your customers and take pressure off the team as there isn't a Release Day anymore.

1. **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.

1. **Describe an efficient workflow for continuous integration?**

CI is a development practice where members of a team integrate their work frequently, with each integration being verified by an automated build to detect errors as quickly as possible.



1. **What are the best practices for DevOps implementation?**

* Agile project management

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Agile teams focus on delivering work in smaller increments, instead of waiting for a single massive release date. Requirements, plans, and results are evaluated continuously, allowing teams to respond to feedback and pivot as necessary.

* Shift left with CI/CD
* Build with the right tools
* Implement automation
* Monitor the DevOps pipeline and applications
* Observability
* Gather continuous feedback
* Change the culture

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

* 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.

* 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.

* 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.

* 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.

* 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.

* 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.

* 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.

* 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.

* 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.
* 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.