**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.

1. What is the difference between GIT and SVN?

The difference between Git and SVN version control systems is that Git is a distributed version control system, whereas SVN is a centralized version control system. Git uses multiple repositories including a centralized repository and server, as well as some local repositories.

1. What are the advantages of using GIT?

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 “Staging Area” or “Index” in GIT?

The staging area is the set of files that have been added to the index, but not committed. Git knows about these files, but they aren't actually part of history (but will be when they are committed).

1. What is GIT stash?

Git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on.

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?
   1. Go into the directory containing the project.
   2. Type git init .
   3. Type git add to add all the relevant files.
   4. You'll probably want to create a .gitignore file right away, to indicate all the files you don't want to track. Use git add . gitignore , too.
   5. Type git commit .
2. 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’?

git remote add just creates an entry in your git config that specifies a name for a particular URL. ... 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?

The git diff is a multi-function Git command, which is used to compare changes committed in Git. Particularly, with the help of this command, you can take two input data sets and output the modifications between them. While executing, this command runs a diff function on Git data source.

1. Explain what the 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.

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

There are couple of reasons to explain why additional commit is necessary:

a) 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.

b) Abusing “git commit- amend” can cause a small commit to grow and acquire unrelated changes.

1. What is Rebasing?

In Git, the rebase command integrates changes from one branch into another. It is an alternative to the better known "merge" command. Most visibly, rebase differs from merge by rewriting the commit history in order to produce a straight, linear succession of commits.

**Maven Fundamentals**

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

Apache Maven, is a software project management, and comprehension tool, based on the concept of a project object model, or POM. M aven can manage a project's build, reporting, and documentation from a central piece of information. ... Maven is designed to provide a simple project setup, that uses best practices as a guide.

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.

1. Explain what is POM and its 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.

1. Explain what a Maven artifact is.

The artifactId element contains the name of the project you are building. In the case if my project name is : Java Web Crawler project, the artifact ID would be java-web-crawler . The artifact ID is used as name for a subdirectory under the group ID directory in the Maven. 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.

1. List out the dependency scope in Maven?

|  |  |
| --- | --- |
| **Sr.No.** | **Scope & Description** |
| 1 | **compile**  This scope indicates that dependency is available in classpath of project. It is default scope. |
| 2 | **provided**  This scope indicates that dependency is to be provided by JDK or web-Server/Container at runtime. |
| 3 | **runtime**  This scope indicates that dependency is not required for compilation, but is required during execution. |
| 4 | **test**  This scope indicates that the dependency is only available for the test compilation and execution phases. |
| 5 | **system**  This scope indicates that you have to provide the system path. |
| 6 | **import**  This scope is only used when dependency is of type pom. This scope indicates that the specified POM should be replaced with the dependencies in that POM's <dependencyManagement> section. |

1. List out what are the build phases in Maven?

Here are some of the most important phases in the default build lifecycle:

* *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, war, …)
* *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

1. 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?

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 repository in Maven holds build artifacts and dependencies of varying types. There are 3 types of maven repository: Local Repository. Central Repository. Remote Repository.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. Maven central repository is located on the web. It has been created by the apache maven community itself. Maven remote repository is located on the web. Most of libraries can be missing from the central repository such as JBoss library etc, so we need to define remote repository in pom.xml file.

1. Explain how you can exclude dependency?
   1. Open the dependency POM and find the transitive dependency you want to exclude.
   2. 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.
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.
* 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 & Agile?

The key differences of Agile and DevOps are as follows:

* 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.

1. What is the need for DevOps?

From planning through delivery, the goal of DevOps is to improve collaboration across the value stream by developing and automating a continuous delivery pipeline. In doing so, DevOps: Increases the frequency and quality of deployments. Improves innovation and risk-taking by making it safer to experiment.

1. What are the advantages of DevOps?

* 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.
* 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.

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.

1. What are the success factors for Continuous Integration?

Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests.Continuous Integration allows you to mitigate risk not only with testing, but also by enabling production parity. Quality Assurance (QA) tasks—such as browser testing—can also be automated, mitigating the risk of a bug making it all the way through to the live site.

1. What are the differences between continuous integration, continuous delivery, and continuous deployment?

Continuous Integration is the practice of testing each change done to your codebase automatically and as early as possible. Continuous Integration and pushes changes to a staging or production system.Continuous Delivery and Continuous Deployment have a lot in common. The key difference is that with Continuous Deployment, your application is run through an automated pipeline workflow. Whereas with Continuous Delivery, your application is ready to be deployed whenever your team decides it's time to do so.

1. What role does the Quality Assurance (QA) team play in DevOps?

QA further enables and directs teams to check the application for performance and quality before it is delivered. The role of QA in the current software development process is changing, where the customer comes into focus and the needs of the customer are brought into perspective while delivering the application.

1. Describe an efficient workflow for continuous integration.
   1. Maintain a code repository
   2. Automate your build
   3. Make your build self-testing
   4. Daily commits to the baseline by everyone on the team
   5. Every commit (to the baseline) should be built
   6. Keep your builds fast
   7. Clone the production environment and test there
   8. Make it easy to get the latest deliverables
   9. Everyone on the team can see the results of your latest build
   10. Automate build deployment
2. What are the best practices for DevOps implementation?
   1. Test Automation.
   2. Integrated Configuration Management.
   3. Integrated Change Management.
   4. Continuous Integration.
   5. Continuous Delivery.
   6. Continuous Deployment.
   7. Application Monitoring.
   8. Automated Dashboards.
3. How will you approach when a project needs to implement DevOps?
   1. First and foremost, it's vital to understand if you really need to implement the DevOps practice.
   2. Once decided, break the culture of organizational silos, and encourage collaboration among the teams.
   3. Bring the focus back to the customer and look to improve end-user satisfaction.
   4. Take small steps and then scale up.
   5. Automate whatever can be automated.
   6. Select compatible tools.
   7. Ensure real-time visibility into the projects.
   8. Integrate and deliver continuously.
   9. Implement robust monitoring and feedback systems.