5.1 Why should companies developing software products use cloud servers to support their development process?

***Cloud server Definition*:** a type of virtual server. Cloud server infrastructure includes powerful resources that allow processing and storing data, applications, information, and even websites. This infrastructure may be completely virtual or physical. In this infrastructure, one (or more) servers are divided into multiple virtual machines (VM) using special cloud software. Therefore, each of these VMs can act as a separate virtual server.

Cloud computing services have enhanced the quality and overall use of computer technology in recent years. Due to the benefits provided by cloud computing services, more developers and companies will use them to manage their technological resources. Cloud systems have provided many benefits, including applications with Lambda functions, more geographic reach, and better-quality websites and servers. ***Some of the benefits are:***

1. Scalability: With traditional server hosting, users need to determine the exact hardware for each system node to run a particular web application or server. With cloud systems, users can now scale any part of an application more quickly and easily.

2. Cost: Another benefit of cloud computing systems is that the cost is much less. Companies and users can save money on computing costs as they will not need any upfront investment and can pay as they go to get the resources they need.

3. Quick Provisioning of Resources: Developers can now quickly create new environments for websites, databases, and virtual machines. They are also able to get additional services that are always being expanded.

4. Multiple Service Models: Cloud provides developers and companies with different options to set up their environment to meet users' unique needs. These service models include Platform as a Service and Infrastructure as a Service. This has been made possible thanks to Item management software.

5. Geographic Reach: Rather than rely on a single location to host software, the cloud allows users to host applications in data centers all over the world.

6. Ease of Deployment: A cloud system enables users to configure automated builds that can deploy certain codes, and databases, automated testing for applications, and automatic provisioning of a particular server.

7. Advanced Services: Cloud can allow developers and companies to use the most advanced services available These services include simplifying microservices and scaling service containers. This results in focusing on building a business's logic instead of managing distributed systems.

8. Monitoring Developer Operations: Developers can benefit from the cloud by more easily monitoring all the operations and applications they use. They can see application performance metrics, logs, and monitoring tools to identify problems with a server or system before a user reports them.

5.2 Explain the fundamental difference between virtualization using a VM and virtualization using a container.

Virtualization is the process of partitioning a physical server into multiple virtual servers. The method of partitioning is carried out using software called 'hypervisor'. After partitioning, the virtual servers act and perform just as a physical server.

Containerization is the process of bundling the application code along with the libraries, configuration files, and dependencies required for the application to run cross-platform. A containerization is an application-packaging approach where the code is written once and capable of executing anywhere, making the application highly portable. Docker is an example of a containerization platform.

Container: The single package of software built using containerization technology is called a 'Container'. The container is a standalone package, independent of the host operating system. As such, it can then execute across multiple platforms, without any issues.

Difference between Virtualization and Containerization

Key Factor	Virtualization	Containerization
Technology	One physical machine has multiple OSs residing on it and appears as multiple machines.	The application developed in a host environment with the same OS and the same machine executes flawlessly on multiple different environments.
Start-up Time	Higher than containers	Less
Speed of working	VMs being a virtual copy of the host server on its operating system, VMs are resource-heavy, hence slower.	Containers are faster.
Size	Larger	Smaller
The component that Virtualizes and the one being virtualized	Hypervisors virtualize the underlying hardware (use of the same hardware).	Containers virtualize the operating system (use of the same OS).

Cost of implementation	Higher	Lower
Benefits for	IT enterprise businesses	Software developers and in turn IT businesses

9.7 What is regression testing, and why is it important? Explain why automated testing makes regression testing straightforward

A regression test is a system-wide test that's intended to ensure that a small change in one part of the system does not break existing functionality elsewhere in the system. It's important because, without regression testing, it's quite possible to introduce intended fixes into a system that creates more problems than they solve.

Regression testing is important for two reasons:

1. It helps developers find and fix bugs in the software, and it helps ensure that the software works as expected before deployment.
2. Regression testing should be done by all developers, but it can also be done by QA teams or by automated tools such as continuous integration servers.

Automated testing is based on the idea that tests should be executable. An executable test includes the input data to the unit that is being tested, the expected result, and a check that the unit returns the expected result.

- Automated regression testing, as opposed to manual regression testing, can free up testers' time and let them focus on tasks that require more creative and critical thinking. It's not to be confused with retesting.

- Although manual testing does make sense in some contexts, a regression suite will usually grow with time. Eventually, it'll reach a point where it's no longer feasible to manage the regression test cases manually.

- When you build a regression suite in an automation tool, it's a one-time effort, and you can keep reusing it indefinitely. You can also keep adding to it over time, as new functionalities are added to the software.

- It is important, however, to select an automation tool that will make it easy for you to maintain a clear overview of your test suite so that it doesn't get messy and unmanageable. You can read more about this in our blog post on how to choose your automation tool.

- Another benefit of automation is that you can schedule your test runs, providing you with immediate feedback when tests fail. With an automation tool that creates visual recordings and logs, you can easily go in and detect why the test failed.

- On a high level, the greatest benefit of automated regression testing, and automated testing in general is probably that it frees up resources. You can set up automation to check specific parts of the software with great accuracy, and then spend brainpower on other types of tests and bug fixes, that will overall enhance your software product and ultimately give your customers a better, and significantly less buggy, experience.

9.8 Explain why it is essential to have a refactoring stage in the test-driven development process.

Refactoring: It is a scientific process of taking existing code and improving it while it makes code more readable, understandable, and clean. Also, it becomes very handy to add new features, build large applications, and spot & fix bugs.

Refactoring also removes "Code Smells" from your project, this is done to get certain benefits and these benefits may not be consumable immediately but over the long term. It is an activity that is a solution to your problems, it's performed when modifying the existing code of a project to incorporate new features or to enhance.

types of Refactoring*:*

Code Refactoring: Simply known as Refactoring, this is the refactoring of source code, it includes (Rename method, Encapsulated field, Extract class, Introduce assertion, and Pushdown method). It changes the structure of a program, but the functionality is the same.

Database Refactoring: A simple change to a database schema that improves its design while re-tuning both its behavior and semantics such as (Rename column, Split the table, moving method, Replacing LOB with table, and Introducing column constraint)

User Interface Refactoring: A simple change to the UI retains its semantics such as (Align entry field, applying common button size, applying font, indicating format, Reword in active voice, and Increasing color contrast) It delivers consistency for all users -both within your organization and your customer's organization.

Reasons why Refactoring is Important:

1. To Keep Your Code Clean: Code refactoring removes all the notions of code smell and makes your code cleaner and easier to understand. You remove redundant code, unnecessary variables, too many parameters, longer methods, longer classes, too many conditions or

unnecessary loops, etc. You clean up all the mess from your code and you eliminate all the defects before it makes real damage.

2. To Improve the Performance of The Application: An application that doesn't have unnecessary classes, functions, variables, methods, or any other mess, runs faster and smoother. Performance of an application increases if the code is recently refreshed or updated. Your application generates quick responses and users no longer complain about the slower performance. This leads to a better customer experience.

3. You Save Time and Money in The Future: A clean and clear code takes less time to understand and implement the new features. None of the developers loves to work or waste their time understanding some messy code. An application that is not refactored takes more time to extend or upgrade. Also, if the application is damaged due to some issues, the organization needs to spend money on fixing the issues and the budget gets increases.

4. To Reduce the Technical Debt: The cost of any software is not finalized when you launch the first version of it. Your software may stop responding after a couple of months if you don't make regular updates to it. You may end up with technical debt and to reduce this debt you need to refactor the code all the time.

5. Your Code is Outdated: Often in development, we use some libraries or frameworks that needs to be updated with time. When a newer version of these libraries or frameworks appears, the program written in the older version may not work or it may work with some errors. If your application uses some libraries that are no longer maintained or even exist, then it can create a lot of problems in the application. Your application may stop responding or you may find a lot of bugs. You need to keep your code up to date to prevent this issue.

6. Makes Bugs Easier to Be Found: It's easy to find bugs in an application when you understand the code and entire structure of your application. You can test the application and find the bug if you understand how things are connected and working in an application.

7. Improves the System Design: You learn new things over time when you improve the code and understand your project in a better way. You implement some features and after a couple of months, if you just pay attention to your code, you will find that it can be written more simply and easily. The solution you implemented months ago cannot be good today. The overall design of your application can be improved by following some best practices and refactoring your code.