TestContainers :

Testcontainers is a Java library that allows you to run tests using Docker containers as lightweight, throwaway instances. It provides a convenient way to spin up and manage the lifecycle of containers during the test execution.

You can use Testcontainers to test applications that have external dependencies, such as databases or message brokers. It also allows you to test applications in multiple runtime environments, such as different versions of a database or operating system.

Testcontainers supports a variety of containers, including databases, message brokers, and web servers. It provides a simple API for starting and stopping containers, as well as executing commands inside the container.

Overall, Testcontainers is a useful tool for improving the reliability and reproducibility of your tests by running them in a consistent, isolated environment.

Testcontainers is a library that allows you to run tests using Docker containers. It provides a convenient API for starting and stopping Docker containers, as well as executing commands inside the container, during the test execution.

By using Testcontainers, you can test applications that have external dependencies, such as databases or message brokers, in a consistent and isolated environment. This can improve the reliability and reproducibility of your tests, as they will be running in a known environment rather than relying on the local system configuration.

Testcontainers supports a variety of different containers, including databases, message brokers, and web servers. It is easy to use and can be integrated with a variety of testing frameworks, such as JUnit or TestNG

Testcontainers for Java is a Java library that supports JUnit tests, providing lightweight, throwaway instances of common databases, Selenium web browsers, or anything else that can run in a Docker container.

Testcontainers make the following kinds of tests easier:

Data access layer integration tests: use a containerized instance of a MySQL, PostgreSQL or Oracle database to test your data access layer code for complete compatibility, but without requiring complex setup on developers' machines and safe in the knowledge that your tests will always start with a known DB state. Any other database type that can be containerized can also be used.

Application integration tests: for running your application in a short-lived test mode with dependencies, such as databases, message queues or web servers.

UI/Acceptance tests: use containerized web browsers, compatible with Selenium, for conducting automated UI tests. Each test can get a fresh instance of the browser, with no browser state, plugin variations or automated browser upgrades to worry

about. And you get a video recording of each test session, or just each session where tests failed.

Much more! Check out the various contributed modules or create your own custom container classes using GenericContainer as a base.