Exercise 8

Task 1: Virtual Machine (VM) vs. Docker

Virtual Machine (VM)	Docker Contanier
Definition: A VM is a complete virtual computer running on top of a physical machine. It uses a hypervisor (like VMware or Hyper-V) to emulate hardware and run a full guest operating system separately from the host. **Structure:** - Physical hardware - Host operating system - Hypervisor - Guest operating system (with its own kernel) - Applications inside the VM **Resource consumption:** High, each VM includes its own OS, so it uses more CPU, memory, and storage. **Isolation level:** Very strong; VMs are fully separated from each other and from the host at the hardware level. **Startup time:** Slow, usually takes minutes to boot up. **Best suited for:** Running different operating systems on the same machine or when strict isolation is required, like running Linux and Windows side by side.	**Definition:** Containers are lightweight, isolated environments that run applications using the host system's kernel. They package an app with its dependencies but don't include a full OS. **Structure:** - Physical hardware - Host operating system + container engine (e.g., Docker) - Containers (each with its own libraries and configuration, but **sharing the host kernel**) - Applications inside the containers **Resource consumption:** Low, since containers don't bundle an entire OS. **Isolation level:** Good, but not as complete as VMs, because they separate processes using kernel features instead of full hardware emulation. **Startup time:** Very fast, typically seconds or less. **Best suited for:** Quickly deploying applications, running many isolated services on one server, and implementing microservices architectures.

Task 2:

The correct answer is: c) frequently integrate code into a shared repository and run automated tests

The main purpose of Continuous Integration (CI) is to help developers merge their changes into a shared codebase frequently (often multiple times a day). Each integration is verified by automated builds and tests, which helps detect errors early, reduce integration problems, and improve code quality.

Task 3: in IntelliJ

Task 4:

The correct answer is: b) Jenkins

Jenkins is one of the most widely used tools for orchestrating CI/CD pipelines. It automates building, testing, and deploying applications, and supports plugins for integrating with many

Task 5:

1.**Parallelize tests**: Split your test suite across multiple executors or runners to run tests simultaneously.

2. **Run only impacted tests (test selection)**: Use tools or plugins to run tests related only to the code changes (e.g., Jest's --changedSince or Bazel's test selection).

３.**Optimize and refactor tests**: Identify and fix slow or flaky tests. Mock expensive external dependencies (like databases, APIs).

4.**Use faster test frameworks**: Some frameworks have faster startup or execution times.

5. **Use build/test caching**: Tools like Gradle, Bazel, or GitHub Actions' caching can reuse results from unchanged parts of your codebase.