- CI/CD- **Continuous Integration** is the practice of frequently integrating code changes into a shared repository, often multiple times a day. Each integration is automatically verified by building the code and running automated tests to detect integration errors or conflicts early in the development process. **Continuous Deployment** is the practice of automatically deploying code changes to production.
- **Agile methodologies** are a set of principles and practices for iterative and collaborative software development. It's designed to help teams plan, track, and manage their work efficiently. Jira supports both Scrum and Kanban methodologies for agile project management. Jira integrates with a wide range of tools and services commonly used in software development and project management, including development tools like GitHub, Bitbucket, and GitLab, as well as communication tools like Slack and Microsoft Teams.
- **Jira:** Jira is a widely-used project management and issue tracking software
- **Docker:**

  Docker is a platform that allows us to package, distribute, and run applications as lightweight, portable containers. Containers are self-contained environments that include everything needed to run an application, such as code, runtime, system tools, libraries, and settings.
- **Kubernetes (often abbreviated as "K8s"):**

  Kubernetes is an open-source container orchestration platform. It automates the deployment, scaling, and management of containerized applications across clusters of hosts.
- **Scrum:** Scrum and Kanban are both popular methodologies used in agile software development, but they have different approaches.Scrum is an iterative and incremental agile framework for managing projects. Scrum follows a set of events, including sprint planning, daily stand-up meetings, sprint review, and sprint retrospective.
- **Kanban:** Unlike Scrum, Kanban does not have fixed-length iterations or predefined roles and events. Instead, it focuses on a continuous flow of work, allowing teams to pull new work into the system as capacity becomes available

- **REST API** (**Representational State Transfer).:** A RESTful API is an architectural style for designing networked applications, particularly web services, that follows a set constraints. RESTful APIs use standard HTTP methods (GET, POST, PUT, DELETE, etc.) to perform operations on resources.
- **Representation**: Resources in a RESTful API are represented in a format such as JSON (JavaScript Object Notation) or XML (eXtensible Markup Language). Clients and servers communicate by exchanging representations of resources.

Here are the key constraints of RESTful APIs:

1. **Client-Server Architecture:** The system is divided into client and server components that are independent of each other. This separation is useful for more flexibility and scalability as both components can evolve independently.
2. **Statelessness:** Each request from the client to the server must contain all the necessary information to understand and process the request. The server does not store any client state between requests.
3. **Cacheability:** Responses from the server should be explicitly labeled as cacheable or non-cacheable. Cacheable responses can be reused by clients that reduces the need for repeated requests to the server and improving performance.

4. **Layered System**: A client should not be able to distinguish whether it is directly connected to the server or communicating through an intermediary (e.g., a proxy server or load balancer). This constraint allows the additional layers for security, load balancing, or caching.

- **Microservices** is an architectural approach to software development where a complex application is built as a collection of small, independent services that communicate with each other through well-defined APIs.