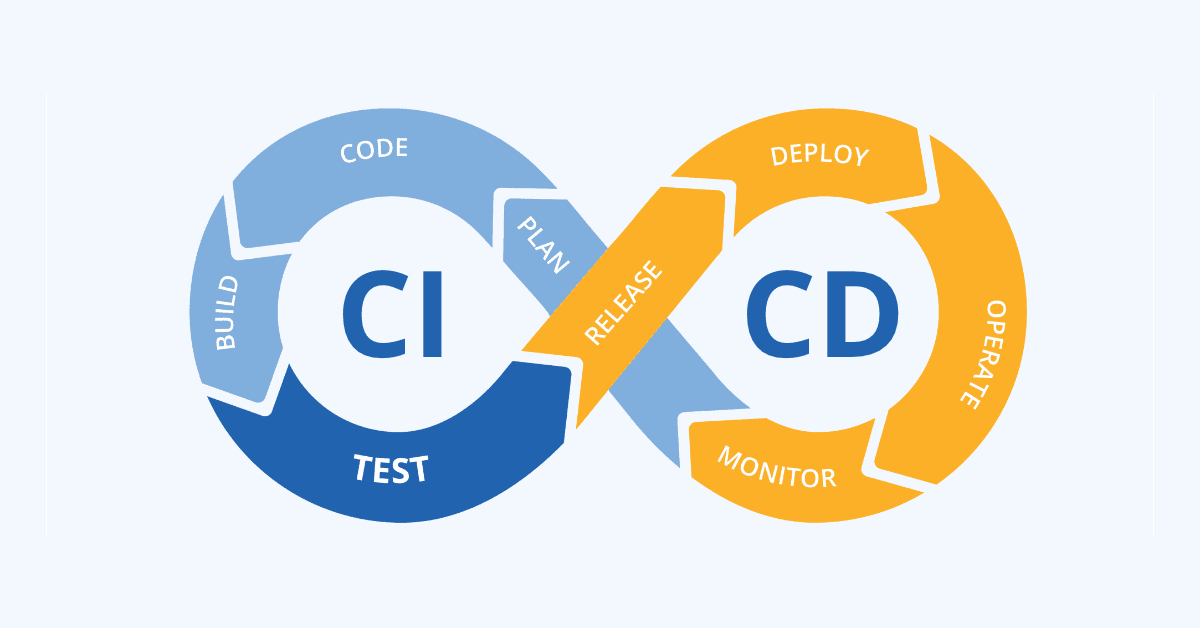
CI/CD Pipeline.



The pipeline me and my group have built, can be divided into three parts.

1. Docker containers for the different parts and a remote REPO.
2. A backend configured for deploying on a standalone tomcat server.
3. A Jenkins pipeline to handle the testing and deployment of the backend.

The Database and the frontend have been provided by the school.

# Docker containers.

There are four docker containers. Frontend, Backend, Server (postgres) and Jenkins container.

The four containers are connected to each other via a docker network, this network is automatically set up when we are creating the containers in the same docker-compose.yml file. There are other ways to connect containers after they have been created but it is easier to create them in the same file and have them connected. The containers will reach each other via their name and the exposed port. The name of the container will work as a hostname.

Example: url: jdbc:postgresql://database:5432/saltmerch

This is the URL that the backend use to connect to the server, and the hostname of the database and the name of the container that servs the database is: database.

The frontend container contains the frontend program and servs it on port 3110. The backend container contains the backend and the standalone tomcat server. The server container contains the database. And the Jenkins container contains the Jenkins program and the running Jenkins pipeline.

# Backend made for standalone tomcat server.

The backend has been reconfigured for a standalone tomcat server. Instead of running on an imbedded tomcat server. The things we had to do in the backend was important but small. We are packaging the program in a war-file instead of a jar-file, and the backend is talking to the same database but in a four-container network instead. We are also using Jenkins to test and deploy the latest update to the backend.

# Jenkins pipeline.

The Jenkins pipeline is the core of the deployment part of the CI/CD pipeline.

The Jenkins pipeline are only handling the backend deployment because of the assignment specification, but I assume that the next step is to set up a pipeline for the frontend to. But in this assignment, we will keep to the backend.

The Jenkins pipeline and the Jenkins program is an effective way of running all the test and deploying a new version of a program. With the Jenkins pipeline we are looking for new updates on the program at pre-decided time stamps. There are methods in Jenkins to let the remote REPO tell Jenkins when there is a new push to the repo, but we are using a timer function. The Jenkins pipeline we have been set up to handle: Building / compiling, running the test, packaging to a war-file, deploying, and generate a test report and store the history of the previous tests.

The stages we have in the pipeline is:

* Clean and checkout = mvn clean -f ./backend/ and git credentialsId:
* Build = mvn test-compile -f ./backend/

Comment: we compile the test but we do not run them.

* Test = mvn surefire: test -f ./backend/

Comment: We are running the goal of the maven phase (Test)

* Package = mvn war:war -f ./backend/

Comment: We are packaging the program in to a war file.

* Deploy = cp backend/target/ROOT.war /artifacts

Comment: We are copying the war file (ROOT.war) to a volume that the jenkins container shares with the backend container.

Post action = junit 'backend/target/\*reports/\*\*/\*.xml'

By Frode Lien 2023-02-01