CI/CD Process - Utilizing Jenkins, Docker, Jest and React.Js (Initial Setup)

This is just to speed up the process for our in class assignment.

Required downloads and setup:

- Docker (https://www.docker.com/products/docker-desktop/). **Create an account as well.**
- Node.js (https://nodejs.org/en/download)
- Visual Studio Code (Or an editor you are comfortable with using.)
- Git Account (Github, Bitbucket, Gitlab, etc)

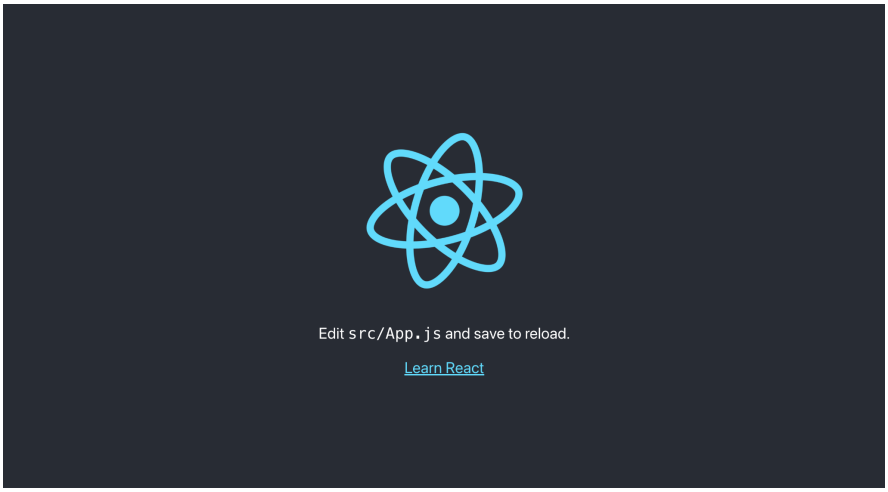
Helpful Videos (Optional)

What is Jenkins?
What is Docker?
What is Git?

Let's create our Simple React Project

Open up your terminal and choose a directory you want your React Project to live. Once in that directory run the following commands to create your react application:

- npx create-react-app simple-app
- cd simple-app
- npm start



It should start the application on **http://localhost:3000** (if the port is not busy) and provides a building block for creating a React application.

Initial Jenkinsfile

Create a file called **'Jenkinsfile'** within your React app directory. The Jenkinsfile will contain the instructions that will be used to automate the CI/CD process. A tool required to run our React application is **nodejs**. This will make npm accessible throughout the pipeline process. Each instruction is defined and split into stages in the Jenkinsfile. First, the pipeline will run **npm install** to download the needed dependencies.

It will then run **npm test** to make sure the application is running properly.

```
pipeline {
 agent any

 tools {nodejs "node"}

 stages {

   stage('Install dependencies') {
     steps {
       sh 'npm install'
     }
   }

   stage('Test') {
     steps {
```

```
        sh 'npm test'
      }
    }
  }
}
```

This is the basic building blocks of the Jenkinsfile, more will be added later on.

Add Project to Github

Create a **public** repository in Github and place the React code with the Jenkinsfile into the repository.

Documentation on how to do it here. If you never used Github, you may need to setup SSH on your computer to push your code to Github.

You now have the Jenkinsfile and repository ready, let's move on to Docker.

Creating a Custom Jenkins Image with a Dockerfile

(If you are feeling Studious, but can be done in class)

Jenkins is an automation server. It's most often associated with building source code and deploying the results. For many, Jenkins is synonymous with continuous integration and continuous delivery (CI/CD).

Let's set up the **Dockerfile** and put it in the parent folder outside the React folder. Here is how the directory structure should look:

./parent-directory

./simple-app (React app)

Dockerfile (Custom Jenkins Dockerfile)

A **Dockerfile** is a set of instructions used for creating a Docker Image. A Docker Image is a template with instructions used to create a Docker Container. We are trying to create a Jenkins container so let's set up the instructions.

The first instruction is **FROM**. The **FROM** instruction specifies the parent image in which we are building. In this case, we are using **FROM** to extend from jenkins. We will be utilizing an image already created for Jenkins that gets constant updates. The last part "**:lts**" specifies that we want the latest version of Jenkins.

The second line gives the container **root** access.

The third line updates the system to provide the most up-to-date packages and versions for the operating system.

The fourth and last line, will install docker within the container using a curl command.

Docker can now be utilized within the container. It will be used to build our React application and push it to your own Docker registry.

```
FROM jenkins/jenkins:lts
USER root
RUN apt-get update
RUN curl -sSL https://get.docker.com/ | sh
```

In class, we will start by running this custom image to create the Jenkins container and installing the necessary plugins.