Desafio 13 – Pipelines

**Nombre:** Matias O. Riedel

El objetivo del desafío es construir un pipeline que se encargue del proceso de automatización de la construcción del DockerFile. El pipeline debe encargarse de tomar el código de la aplicación, construir una imagen, levantar un contenedor y subir esa imagen a DockerHub.

Estructura de Archivos:

Desafio\_13

App/

--app.py

--requirements.txt

-- tests/

---- test\_app.py

Dockerfile

Jenkinsfile

.github/

--workflows/

----main.yml

README.md

Para lo siguiente procederemos a crear nuestros archivos de la app, donde ingresaremos el código de Python.

Captura de pantalla con la imagen de una pantalla

Descripción generada automáticamente

Editamos nuestros archivo de requisitos (requirements.txt) donde ingresaremos las dependencias de nuestra aplicacion  
Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Procederemos a crear nuestra estructura de test de la app, las que nos servirán para realizar pruebas unitarias de la app, ingresaremos las siguientes líneas de código:

Texto

Descripción generada automáticamente

Crearemos nuestros archivos Dockerfile y Jenkisfile donde armaremos la estructura y los parámetros customizados que necesitamos para construir nuestra aplicación

Texto

Descripción generada automáticamente

Jenkisfile, donde especificaremos la aplicación de nuestro pipeline de jenkins

Imagen que contiene Texto

Descripción generada automáticamente

pipeline {

    agent any

    environment {

        DOCKER\_HUB\_CREDENTIALS = credentials('docker-hub-credentials')

    }

    stages {

        stage('Build Image') {

            steps {

                script {

                    docker.build('myapp:latest', '.')

                }

            }

        }

        stage('Run Linting') {

            steps {

                sh 'docker run --rm -i hadolint/hadolint < Dockerfile'

            }

        }

        stage('Test Application') {

            steps {

                script {

                    docker.build('myapp:test', '.').inside {

                        sh 'pytest tests/'

                    }

                }

            }

        }

        stage('Deploy Image') {

            steps {

                script {

                    docker.withRegistry('https://index.docker.io/v1/', 'DOCKER\_HUB\_CREDENTIALS') {

                        docker.image('myapp:latest').push()

                    }

                }

            }

        }

        stage('Run Container') {

            steps {

                script {

                    try {

                        sh 'docker rm -f myapp-container || true'

                    } catch (Exception e) {

                        echo 'Container not running'

                    }

                    sh 'docker run -d --name myapp-container -p 80:80 myapp:latest'

                }

            }

        }

    }

    triggers {

        cron('H H \* \* \*')

    }

}

Crearemos nuestros directorios de .github donde crearemos nuestro archivo mail.yml para especificar la configuración de nuestro pipeline de github actions de nuestro proyecto:

Texto

Descripción generada automáticamente

Luego de tener todos nuestros parámetros y nuestra estructura armada de nuestro proyecto procedemos a ejecutar el mismo para poder hacer el build:

Texto

Descripción generada automáticamente

Luego de que termine de realizar toda la build de nuestros proyecto ejecutamos con el comando:

Texto

Descripción generada automáticamente