Experiment 4

Docker Build and Push
using GitHub Actions

Aim

Set up a GitHub Actions workflow to automatically build a Docker image from a Dockerfile in your GitHub repository and push it to a container registry (e.g., Docker Hub).

Steps

1. Create a project, Add Dockerfile and Push it on GitHub
   a. dCreate the project

```
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
lazypunk@pop-os:~/Desktop/EXP4_JENKINS$ npm init -y
Wrote to /home/lazypunk/Desktop/EXP4_JENKINS/package.json:

{
  "name": "exp4_jenkins",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

   b. Install the packages

```
lazypunk@pop-os:~/Desktop/EXP4_JENKINS$ npm install epress ejs

added 17 packages, and audited 18 packages in 36s

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

c. Add the server.js file that runs a basic server that renders index.ejs file present inside views folder.

```
lazypunk@pop-os:~/Desktop/EXP4_JENKINS$ ls
Dockerfile  node_modules  package.json  package-lock.json  server.js  views
lazypunk@pop-os:~/Desktop/EXP4_JENKINS$
```

d. Run & test the server

```
lazypunk@pop-os:~/Desktop/EXP4_JENKINS$ node server.js
Server listening on port 3000!
```

HELLO DOCKER

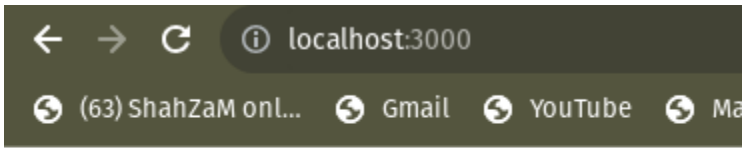
e. Add a Dockerfile & write the configuration to run the project

```
Dockerfile  node_modules  package.json  package-lock.jso
lazypunk@pop-os:~/Desktop/EXP4_JENKINS$ cat Dockerfile
FROM node:lts
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

f. Build & run the image using Dockerfile to test the application

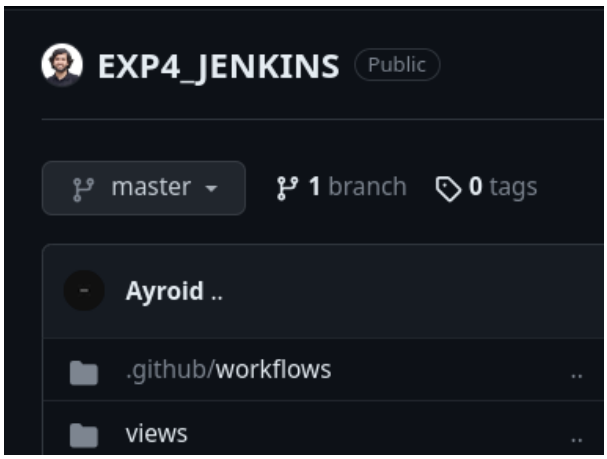
```
lazypunk@pop-os:~/Desktop/EXP4_JENKINS$ docker build -t exp4_jenkins .
[+] Building 107.6s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 152B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/node:lts
 => [1/5] FROM docker.io/library/node:lts@sha256:62efd17e997bc843aefa4
 => => resolve docker.io/library/node:lts@sha256:62efd17e997bc843aefa4
 => => sha256:62efd17e997bc843aefa4c003ed84f43dfac83fa6228c57c898482e5
 => => sha256:42a4d97d4abf2f278c323370cf9c8dbccc2a238acceebceb53711926
 => => sha256:c7f379c300449f790e0f1d7f54c40c3b28825e042d966aca151277f6
 => => sha256:0a9573503463fd3166b5b1f34a7720dac28609e98d731e50e7068f92
 => => sha256:1ccc26d841b4acc2562483bf393ce1cf8bc358ced09ccd8254252268
 => => sha256:800d84653581fc119cd75cd572fa190d3b813d49221b9e5ee463e356
```

```
lazypunk@pop-os:~/Desktop/EXP4_JENKINS$ docker run --rm -it --name exp4_jenkins -p 3000:3000 exp4_jenkins

> exp4@1.0.0 start
> node server.js

Server listening on port 3000!
```



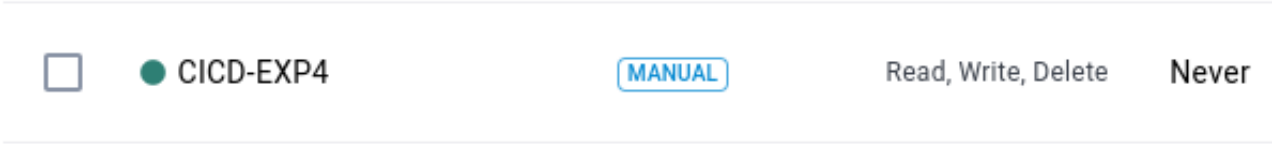
HELLO DOCKER

g. Create a GitHub repository and push the code to it



2. Create a Docker Hub access token

   a. Log in to your Docker Hub account.
   b. Go to your account settings and click on the "Security" tab.
   c. Under "Access Tokens," click "New Access Token." Give it a name, select the required permissions (e.g., "Write" for pushing Docker images), and click "Create."
   d. Copy the generated access token. You will need it to authenticate with Docker Hub in your GitHub Actions workflow.



3. Create a GitHub Actions Workflow
   a. Create a directory named .github/workflows

   b. Create a YAML file inside this directory

c. Write the necessary configuration in it

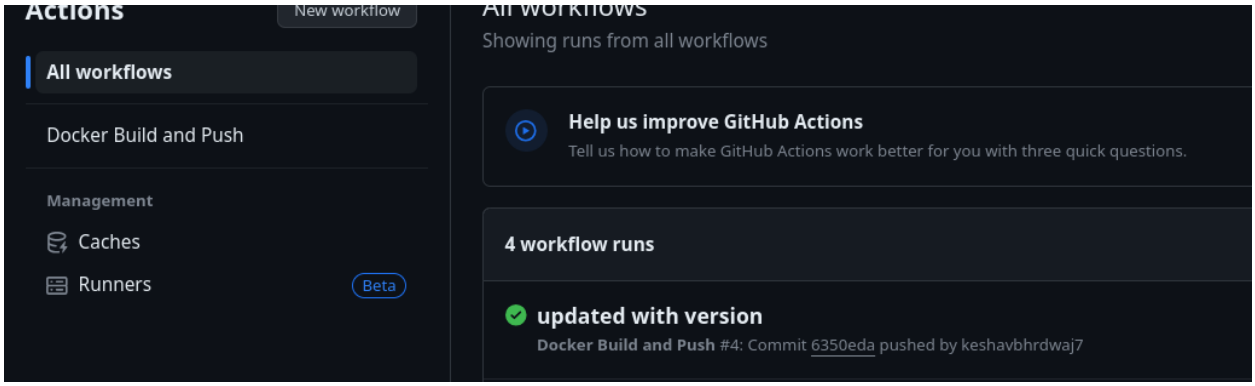
```
lazypunk@pop-os:~/Desktop/EXP4_JENKINS/.github/workflows$ cat docker-build-and-push.yml
name: Docker Build and Push
on:
  push:
    branches:
      - master

jobs:
  build-and-push:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Login to Docker Hub
      run: docker login -u ${{ secrets.DOCKER_USERNAME }} -p ${{ secrets.DOCKER_PASSWORD }}
      env:
        DOCKER_USERNAME: ${{ secrets.DOCKER_USERNAME }}
        DOCKER_PASSWORD: ${{ secrets.DOCKER_PASSWORD }}

    - name: Build and Push Docker Image
      run: |
        docker build -t ${{ secrets.DOCKER_USERNAME }}/CICD-EXP4:v2 .
        docker push ${{ secrets.DOCKER_USERNAME }}/CICD-EXP4:v2
```

4. In the GitHub repository setup the Docker Username and Password as secrets

5. Commit and Push Changes

```
lazypunk@pop-os:~/Desktop/EXP4_JENKINS$ git push
Username for 'https://github.com': keshavbhrdwaj7
Password for 'https://keshavbhrdwaj7@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
```

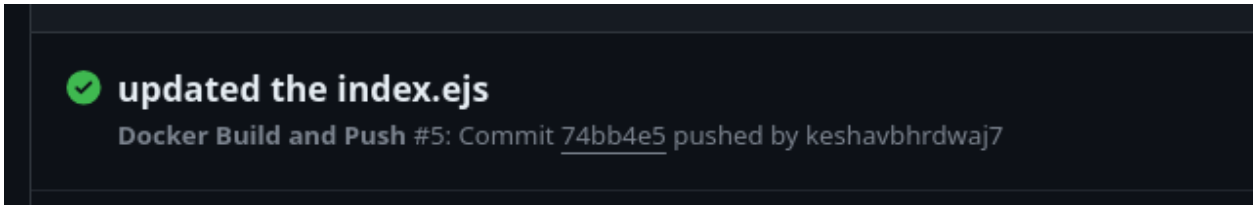
6. Check the Workflow Status

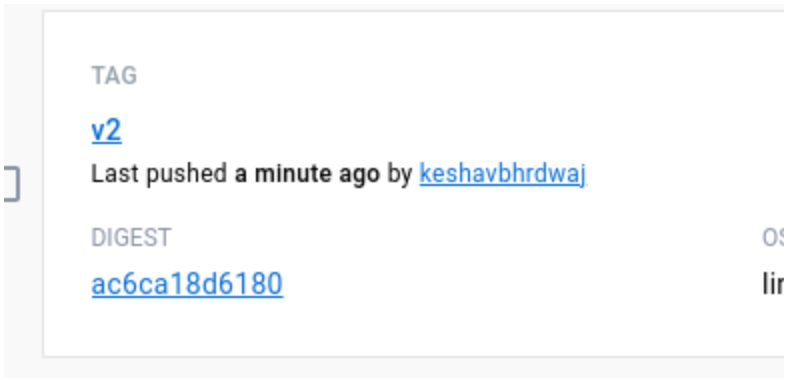


7. Verify the Docker Image on Docker Hub

8. Triggering another build to verify

```
lazypunk@pop-os:~/Desktop/EXP4_JENKINS/views$ git push
Username for 'https://github.com': keshavbhrdwaj7
Password for 'https://keshavbhrdwaj7@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 372 bytes | 372.00 KiB/s, do
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
```





An image with a v2 tag is created on DockerHub.