CONTINUOUS INTEGRATION AND CONTINUOUS DEPLOYMENT LAB

Lab File
(2023-2024)

for

5th Semester

Submitted To

Dr. Hitesh Kumar Sharma
CI/CD Professor, Cluster Head
(Cybernetics)
School of Computer Science

Submitted By:

Arpit Goyal
B. Tech. CSE DevOps [5th Semester]
500094790
R2142210148
B-3

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 yourGitHub 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. Create the project

```
→  exp4 npm init -y
Wrote to /home/ayroid/Documents/Work/College/SEM5/Subjects/cicd/lab/HiteshSir/exp4/package.json:

{
  "name": "exp4",
  "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

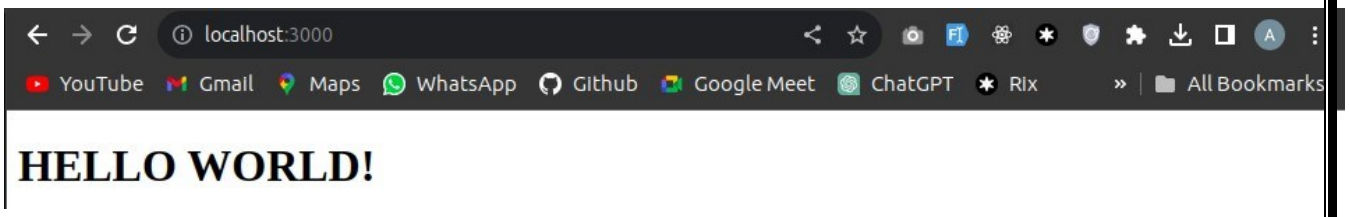
```
→  exp4 npm install express ejs

added 74 packages, and audited 75 packages in 3s

10 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 viewsfolder

```
→  exp4 ls
node_modules  package.json  package-lock.json  server.js  views
→  exp4
```

d. run & test the server

```
→  exp4 nodemon server.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
App listening on port 3000!
```



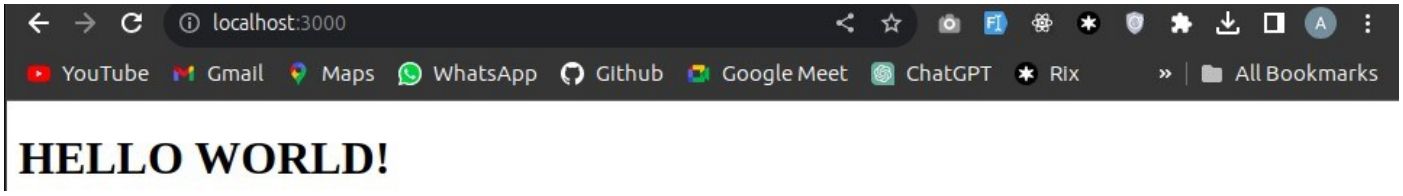
e. Add a Dockerfile & write the configuration to run the project

```
→  exp4 cat Dockerfile
FROM node:lts
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]%
→  exp4
```

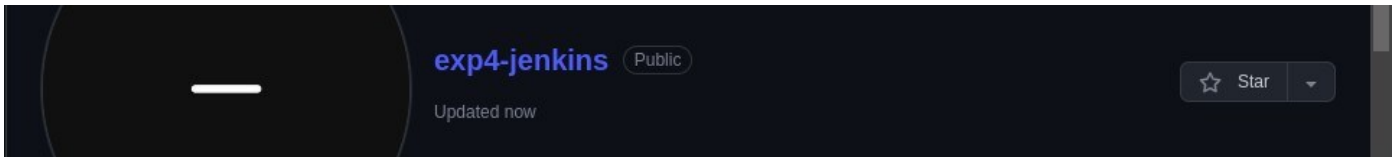
f. Build & run the image using Dockerfile to test the application

```
→  exp4 docker build -t exp4-jenkins .
[+] Building 2.4s (11/11) FINISHED                                    docker:default
 => [internal] load .dockerignore                                             0.0s
 => => transferring context: 2B                                               0.0s
```

```
→  exp4 docker run --rm -it --name exp4-jenkins -p 3000:3000 exp4-jenkins

> exp4@1.0.0 start
> node server.js

App listening on port 3000!
```



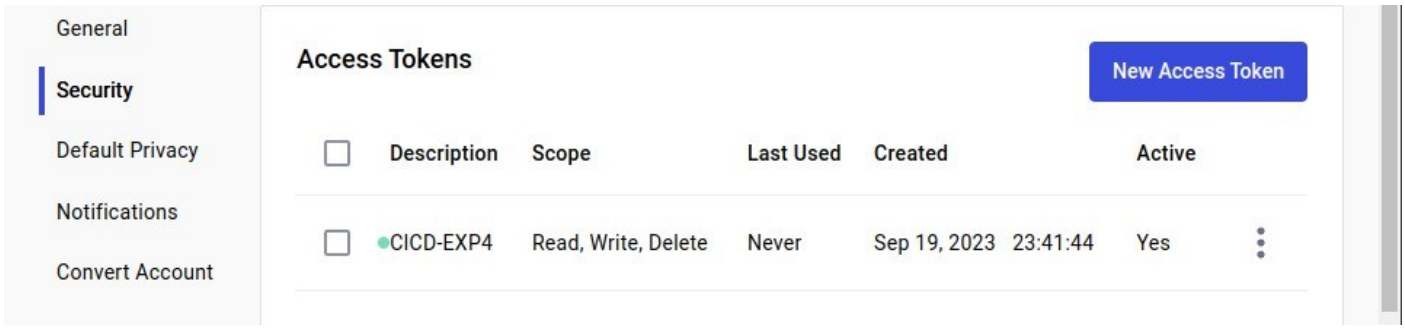
g. Create a GitHub repository and push the code to it




```
→  exp4 git:(master) git add . && git commit -m "Updates" && git push origin master
[master 529c008] Updates
 1 file changed, 1 insertion(+), 1 deletion(-)
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 291 bytes | 291.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Ayroid/exp4-jenkins.git
   0fc4791..529c008  master -> master
→  exp4 git:(master)
```

2. Create 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 requiredpermissions (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 yourGitHub Actions workflow.



3. Create a GitHub Actions Workflow

a. Create a directory named .github/workflows

```
→  exp4 git:(master) mkdir -p ./.github/workflows
→  exp4 git:(master) ls
```

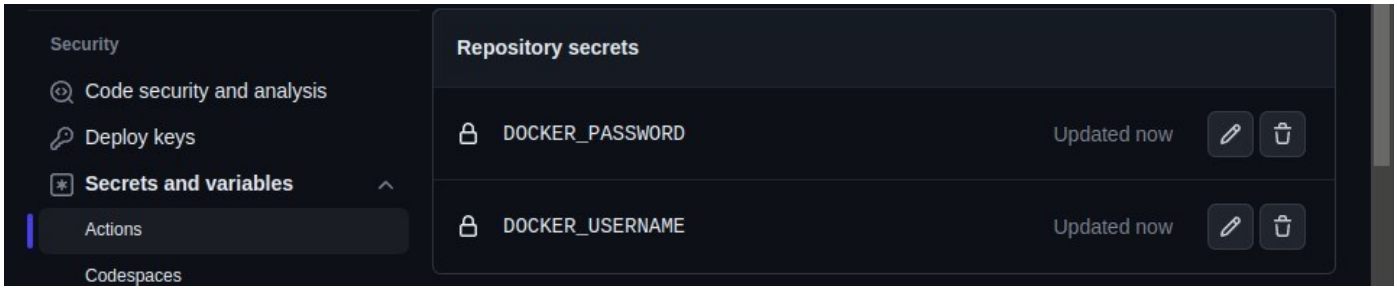
b. Create a YAML file inside this directory

```
→  exp4 git:(master) cd .github/workflows
→  workflows git:(master) touch docker-build-and-push.yml
```

c. Write the necessary configuration in it

```
→  workflows git:(master) ✗ 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:v1 .
        docker push ${{ secrets.DOCKER_USERNAME }}/cicd-exp4:v1
→  workflows git:(master) ✗
```

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

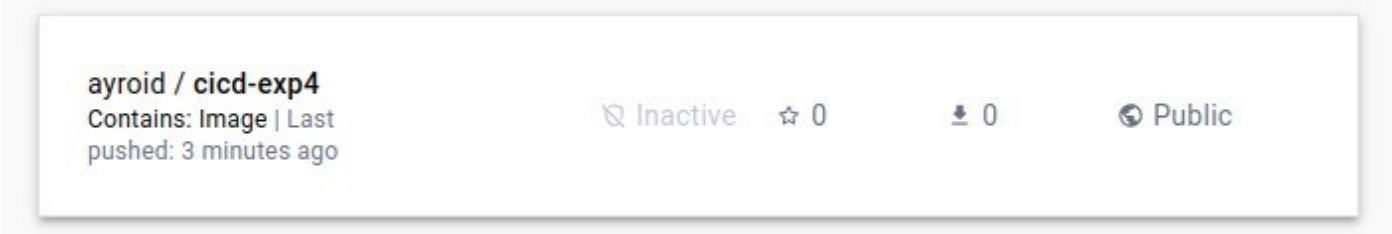


5. Commit and Push Changes

```
→  exp4 git:(master) ✗ gac "Node Version Updated in Dockerfile" && gpo master
[master 9cb12c3] Node Version Updated in Dockerfile
 1 file changed, 1 insertion(+), 1 deletion(-)
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 315 bytes | 315.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Ayroid/exp4-jenkins.git
   6bb2fe1..9cb12c3  master -> master
→  exp4 git:(master)
```

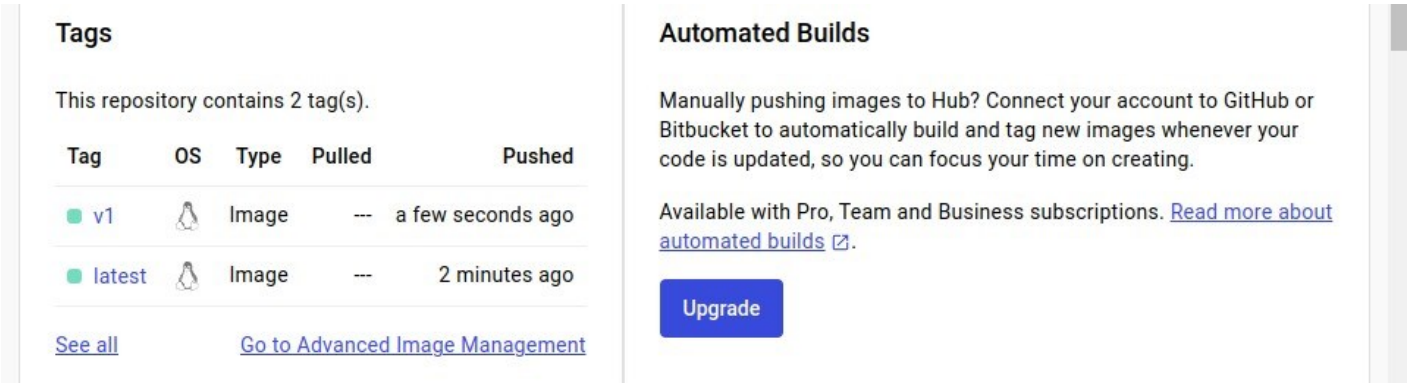
6. Check the Workflow Status

7. Verify the Docker Image on Docker Hub



8. Triggering another build to verify

```
→  exp4 git:(master) gac "Triggering a Build" && gpo master
[master b6534ac] Triggering a Build
 1 file changed, 2 insertions(+), 2 deletions(-)
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 419 bytes | 419.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Ayroid/exp4-jenkins.git
   08b34c3..b6534ac  master -> master
→  exp4 git:(master)
```



An image with v1 tag is created on DockerHub.