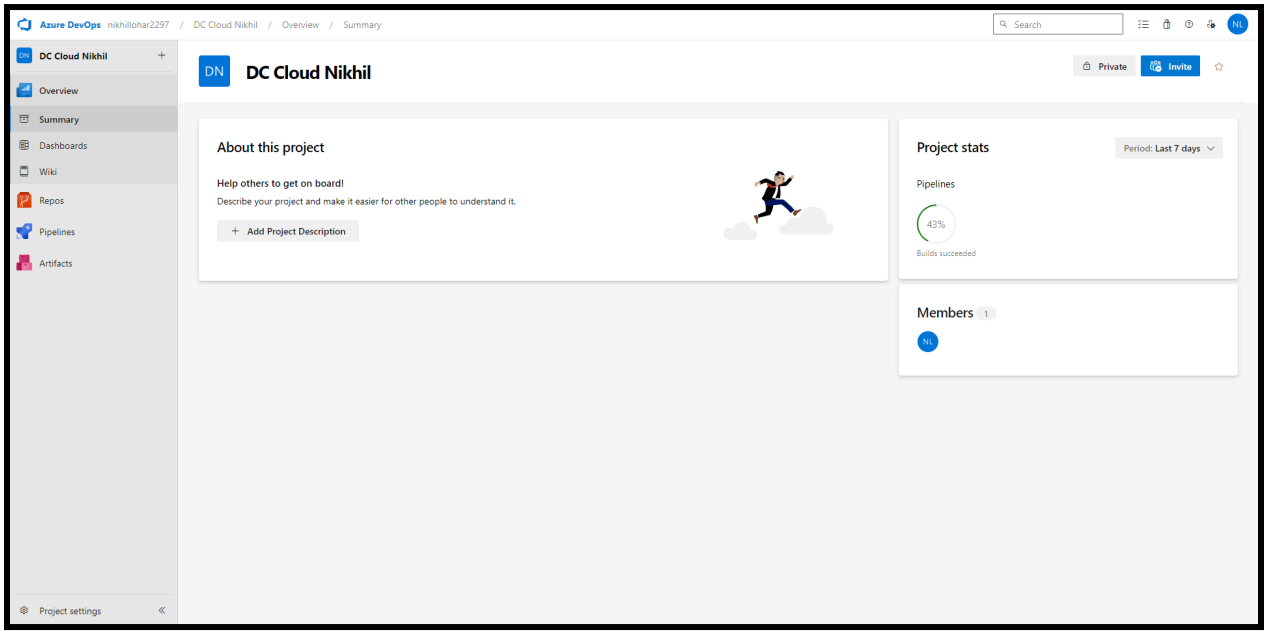
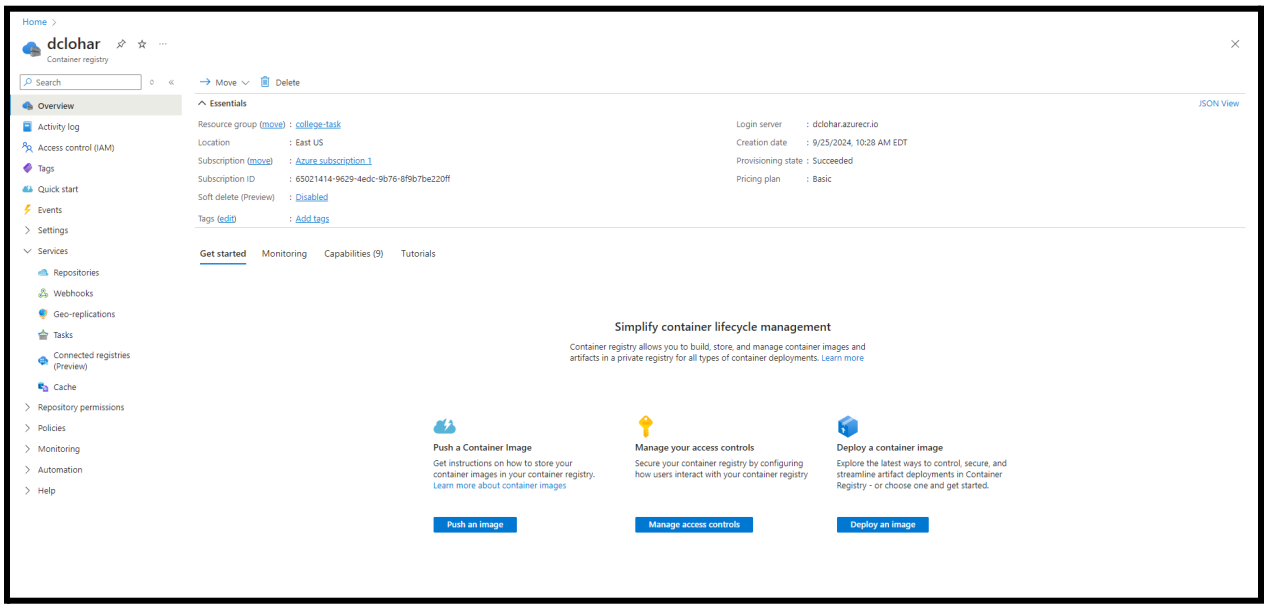
Nikhil Lohar - 100925168

Assignment 2 : Azure Pipeline

Task 1 : Create a New Devops Organization and a private project.

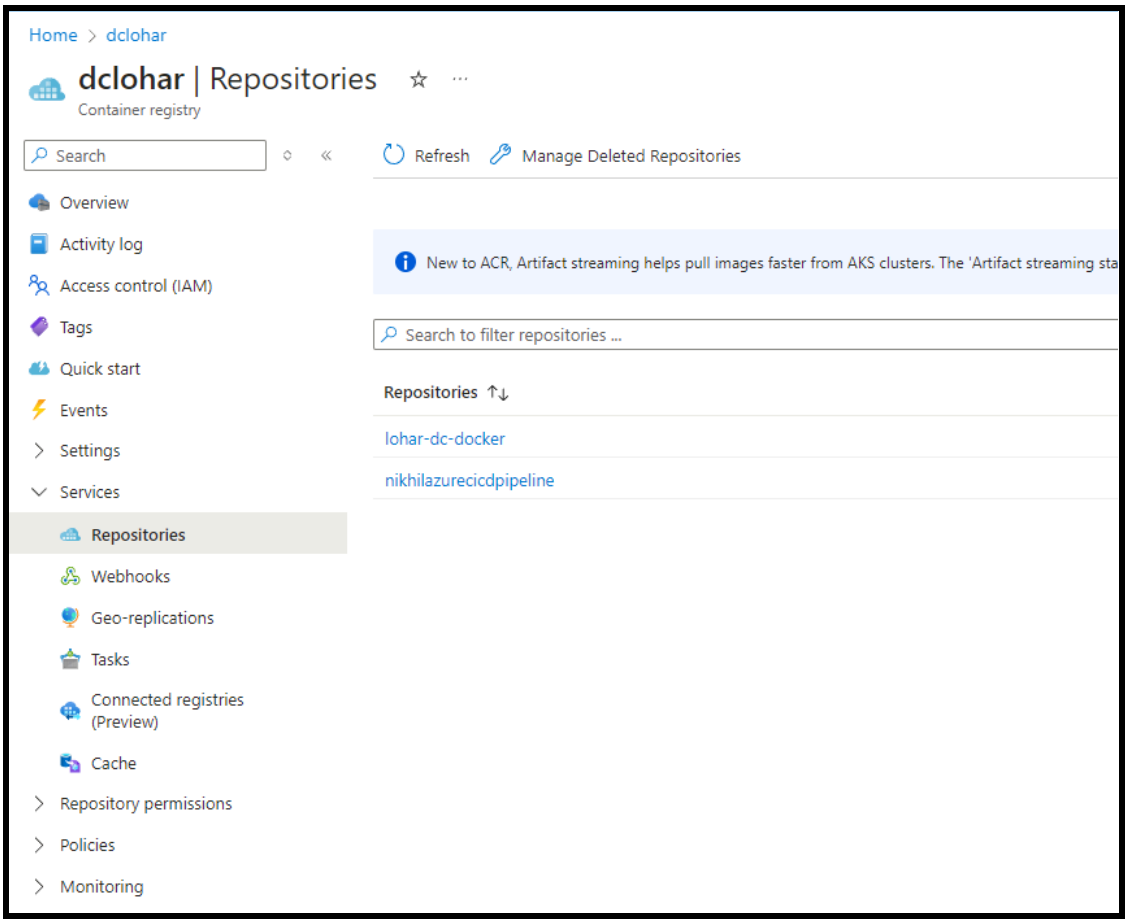


Task 2 : Create a new azure container registry

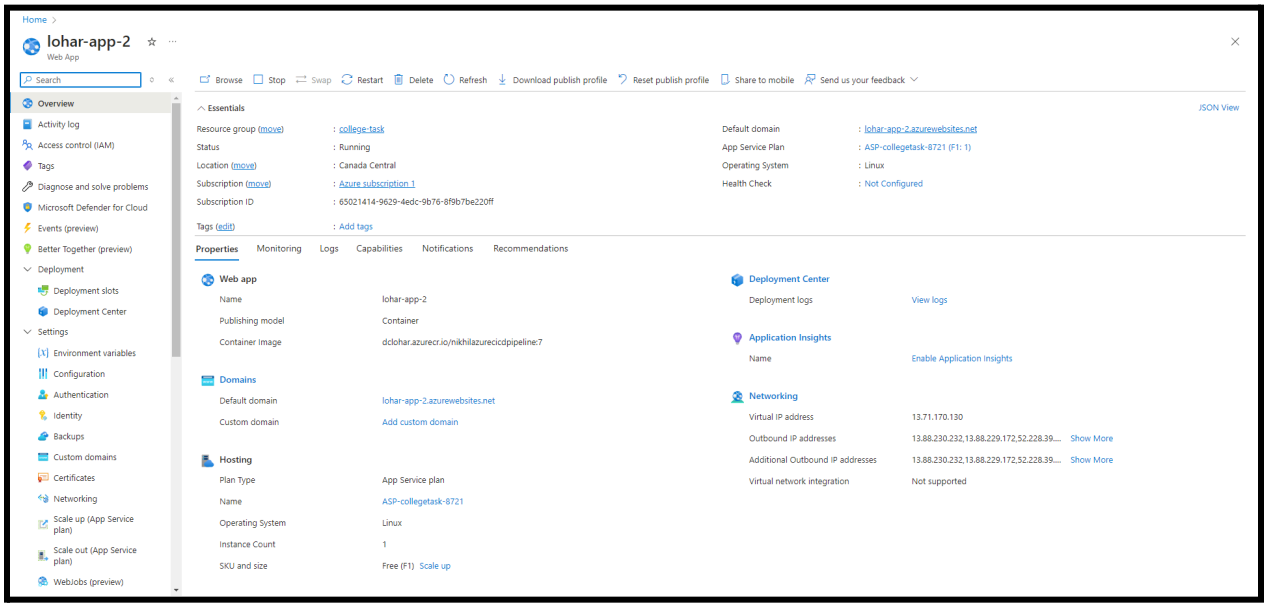


```
PS D:\projects\azure-ci-cd-pipeline> docker build -t dclohar.azurecr.io/lohar-dc-docker:latest .
[+] Building 1.0s (14/14) FINISHED                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 377B                                                                                0.0s
 => [internal] load metadata for docker.io/library/node:22.9.0-alpine                                               0.8s
 => [auth] library/node:pull token for registry-1.docker.io                                                         0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 53B                                                                                    0.0s
 => [1/8] FROM docker.io/library/node:22.9.0-alpine@sha256:c9bb43423a6229aeddf3d16ae6aaa0ff71a0b2951ce18ec8fedb6f   0.0s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 2.20kB                                                                                 0.0s
 => CACHED [2/8] RUN mkdir /azure-ci-cd                                                                             0.0s
 => CACHED [3/8] RUN chown -R node:node /azure-ci-cd                                                                0.0s
 => CACHED [4/8] WORKDIR /azure-ci-cd                                                                               0.0s
 => CACHED [5/8] COPY --chown=node:node package.json /azure-ci-cd                                                   0.0s
 => CACHED [6/8] COPY --chown=node:node package-lock.json /azure-ci-cd                                              0.0s
 => CACHED [7/8] RUN npm install                                                                                    0.0s
 => CACHED [8/8] COPY --chown=node:node . .                                                                         0.0s
 => exporting to image                                                                                              0.0s
 => => exporting layers                                                                                             0.0s
 => => writing image sha256:59a38d7ca62d1876be70a6b82ed5427edd70dc468046f9754ef58e3e334970ec                        0.0s
 => => naming to dclohar.azurecr.io/lohar-dc-docker:latest                                                          0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/sr95wxkt24450rlvuv2zb7lh8

What's next:
    View a summary of image vulnerabilities and recommendations → docker scou
PS D:\projects\azure-ci-cd-pipeline> docker push dclohar.azurecr.io/lohar-dc-docker:latest
The push refers to repository [dclohar.azurecr.io/lohar-dc-docker]
7550aa6fa802: Pushed
2ac640793789: Pushed
0107beb64b1d: Pushed
e9de5febf3d9: Pushed
5f70bf18a086: Pushed
288077570119: Pushed
1488a9c0946a: Pushed
6a8f65b6edec: Pushed
b298ceddbfb8: Pushed
44b1b6f4e77e: Pushed
63ca1fbb43ae: Pushed
latest: digest: sha256:7cf2dc006c3952ffde139fc83c7974b73257fbe83e76e4e2f1c22f201f6cc0a7 size: 2615
PS D:\projects\azure-ci-cd-pipeline>
```

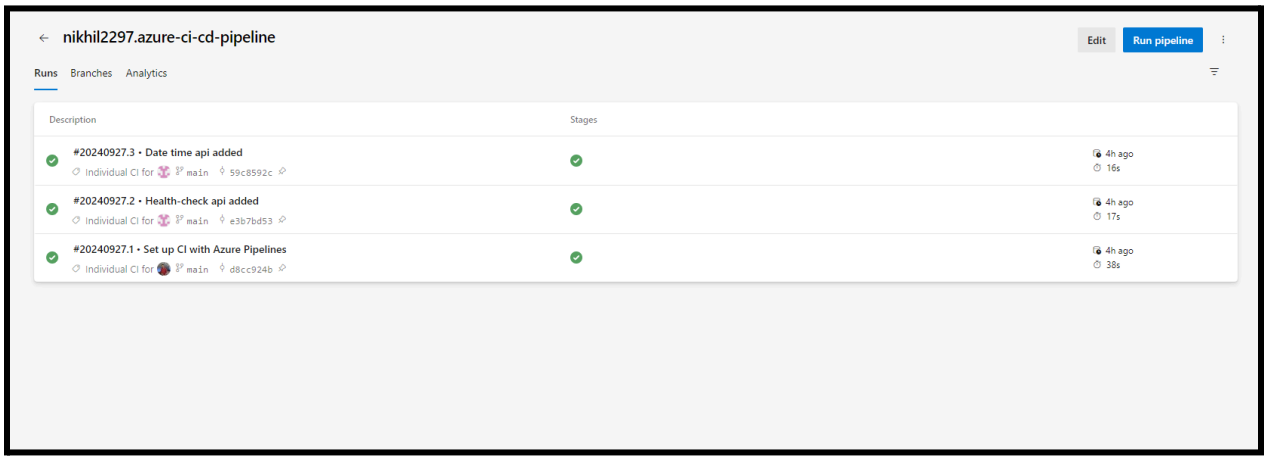
docker-desktop://dashboard/build/desktop-linux/desktop-linux/sr95wxkt24450rlvuv2zb7lh8
Ctrl+Click to follow link

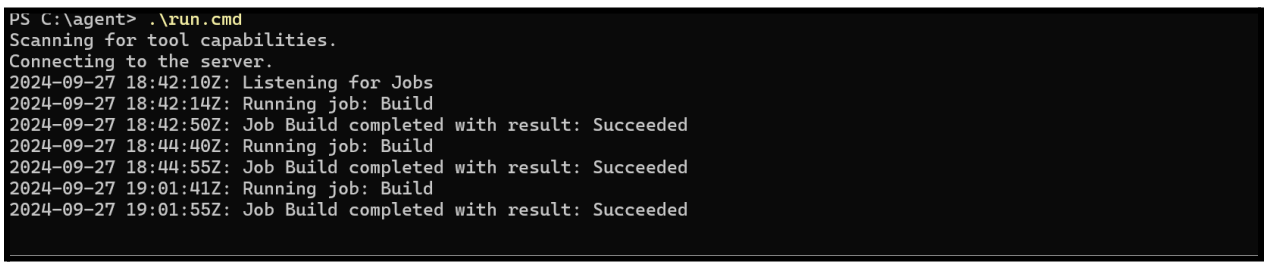


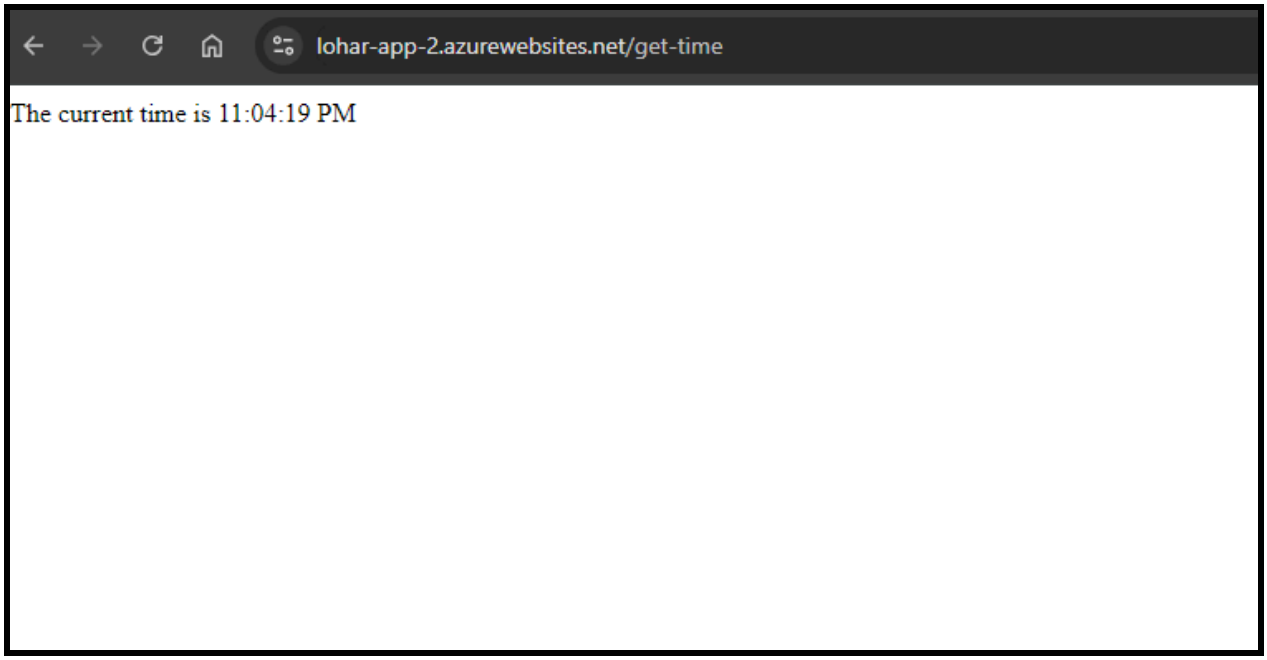
Task 3 : Create a Web App service for Linux containers that will run the application



Task 4 : Azure Pipeline and connect it with github which creates a container and pushes it to the container registry on every commit.







Journal

For this project, I created a Node.js API and hosted it using Azure services with an automated CI/CD pipeline. I first set up a GitHub repository and created a Dockerfile to containerize the application. After testing locally, I created an Azure Container Registry (ACR) and pushed the Docker image to it. I then deployed the app using Azure Web App for Containers, linking it to the ACR. To automate this, I created an Azure DevOps pipeline connected to GitHub, which builds and pushes a new Docker image on every commit. Due to Azure's free tier limitation, I configured my local machine as a self-hosted agent and updated the pipeline to run locally, ensuring that every GitHub commit triggered the pipeline, rebuilt the Docker image, and updated the hosted application. This setup provided full automation for the application's deployment process.