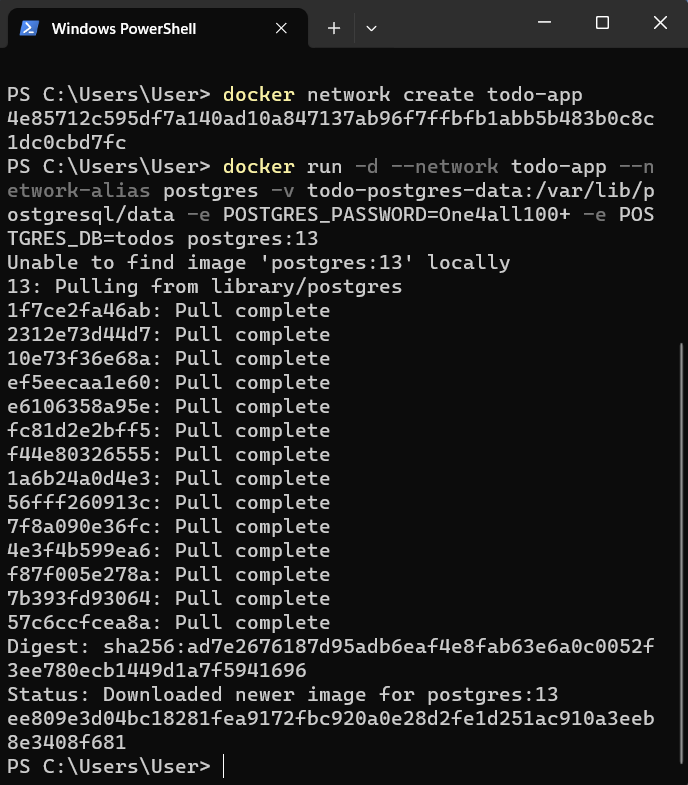
Link to git:

Part 1.

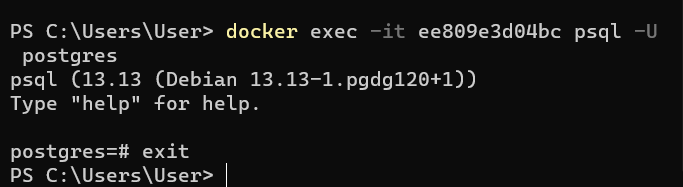
1. Create the network and start a PostgreSQL container and attach it to the network



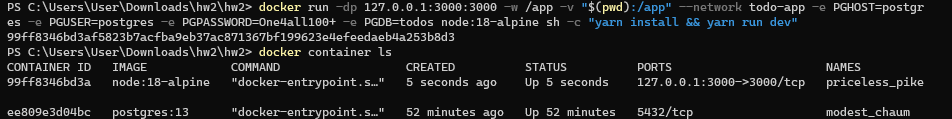
2. Get the container id



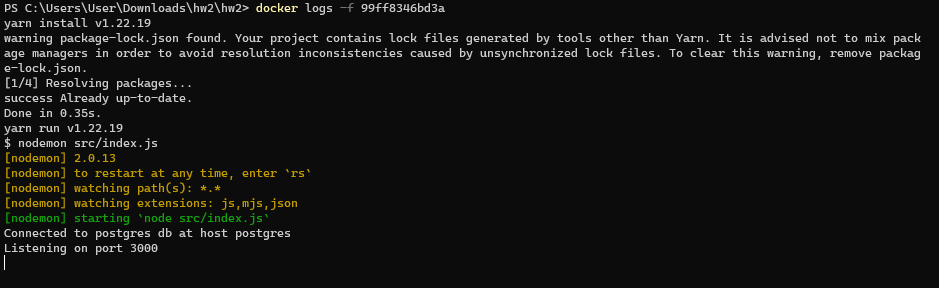
3. Verify the connection to the database



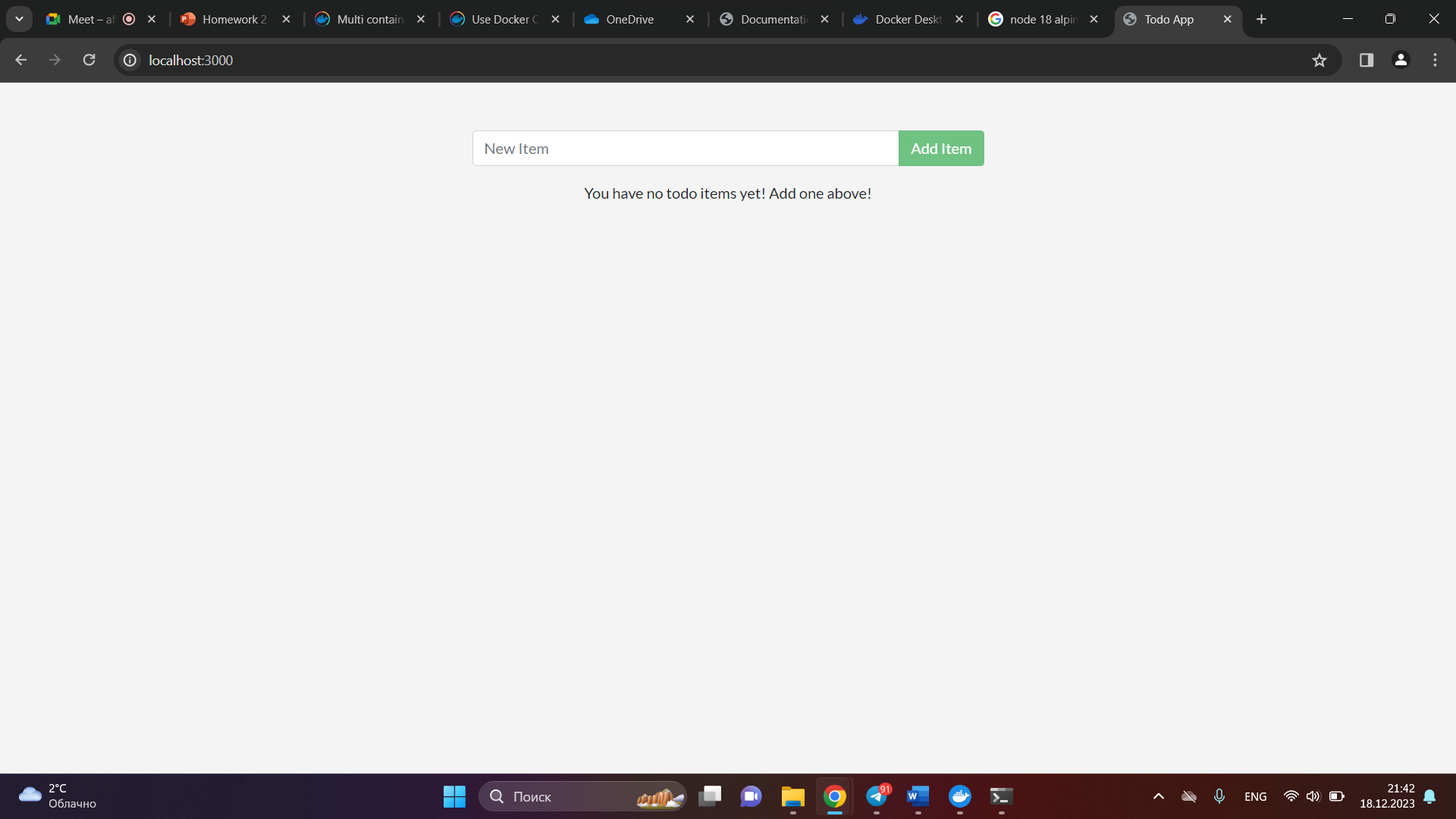
4. Start dev-ready container



5. Examine the logs for the container to see if postgres is listening

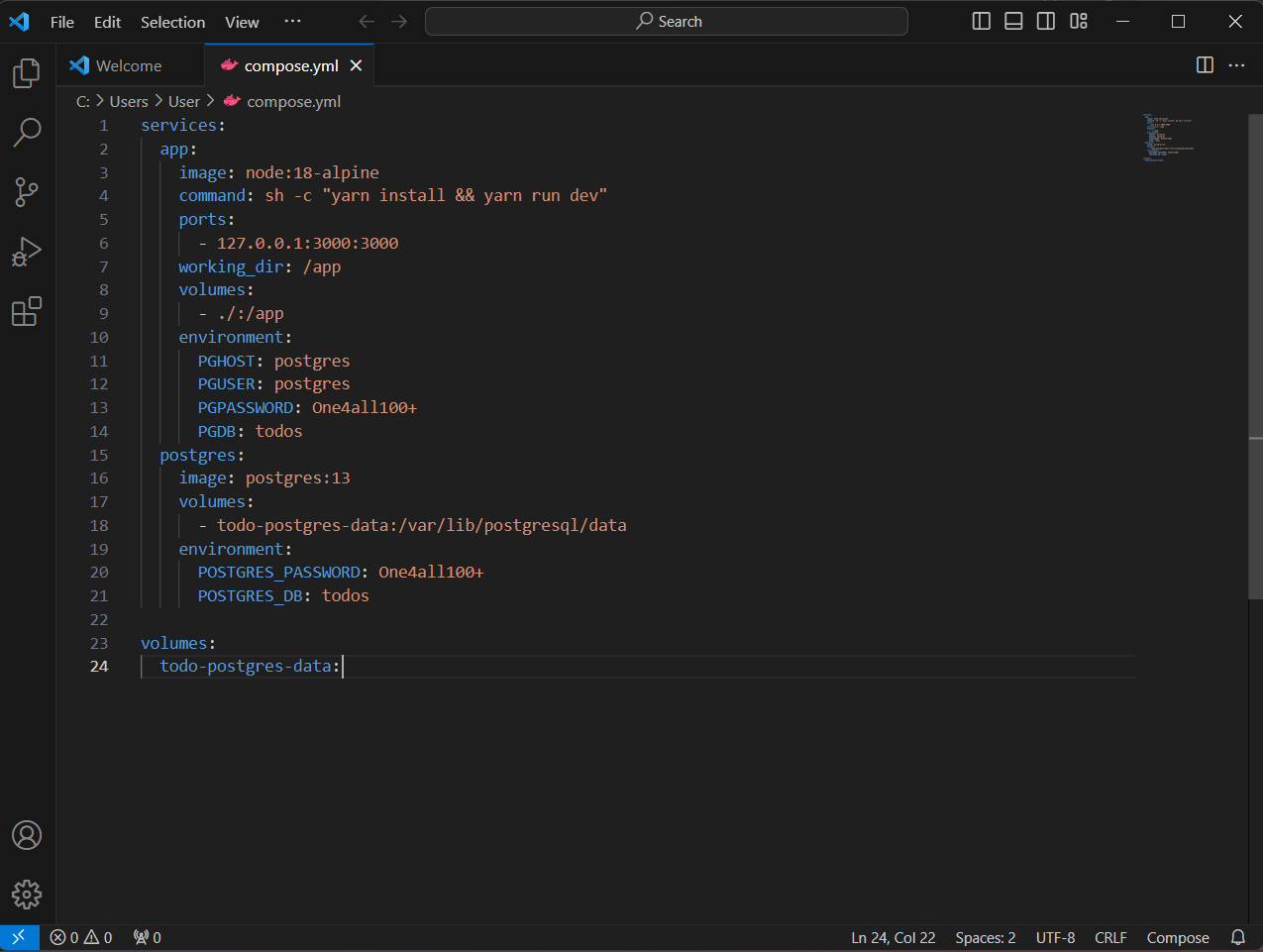


6. It works



7. Removing the container

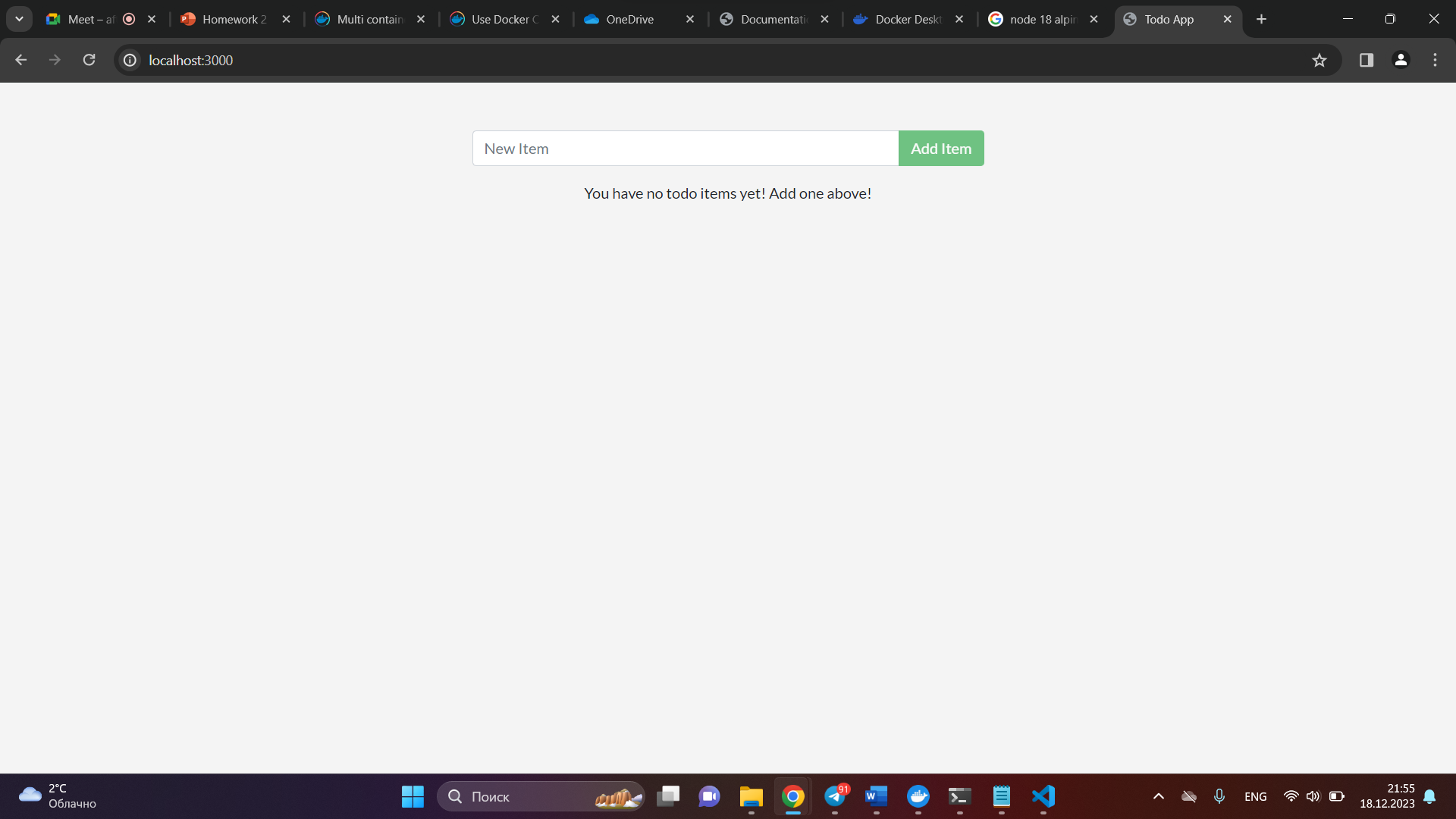
8. Creating the compose.yaml file:



9. Starting up the application stack using the docker compose up command

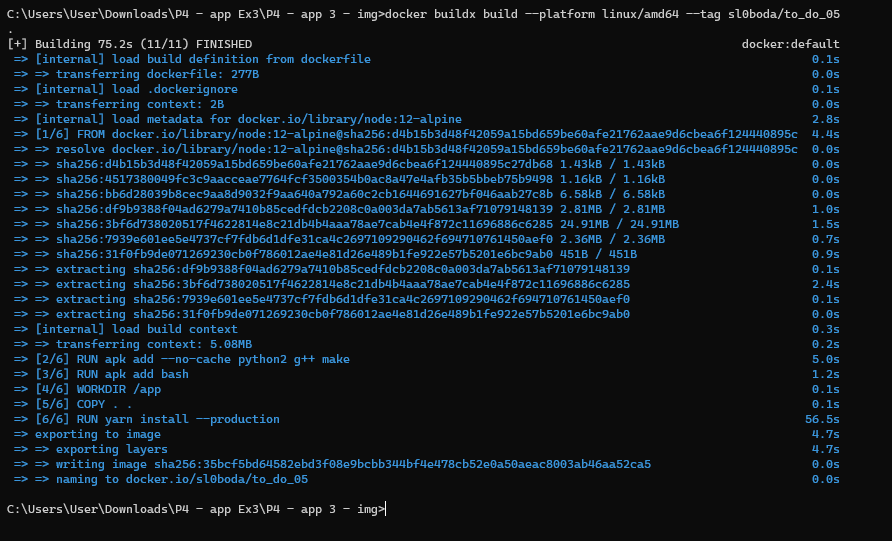


10. Done!

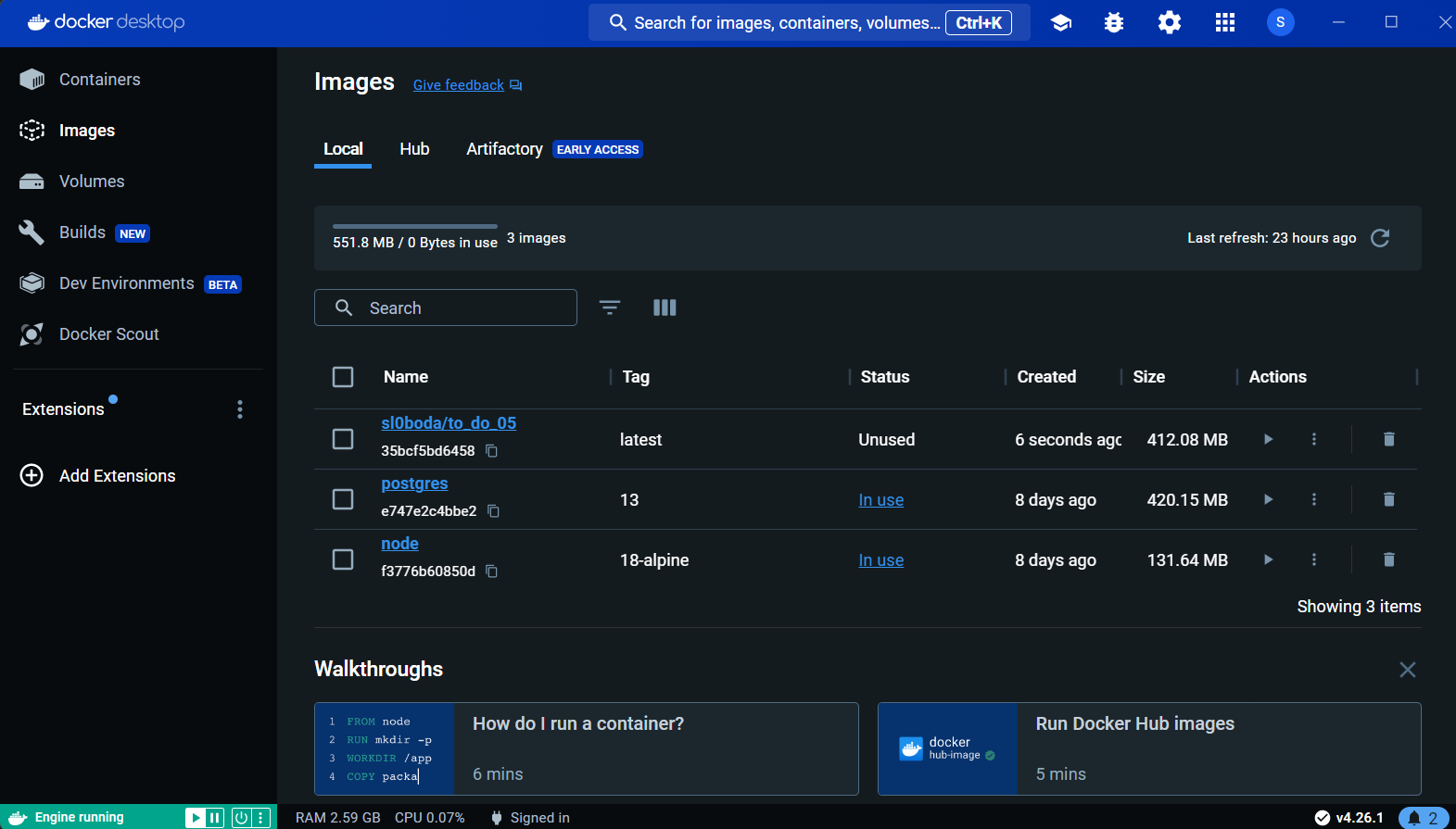


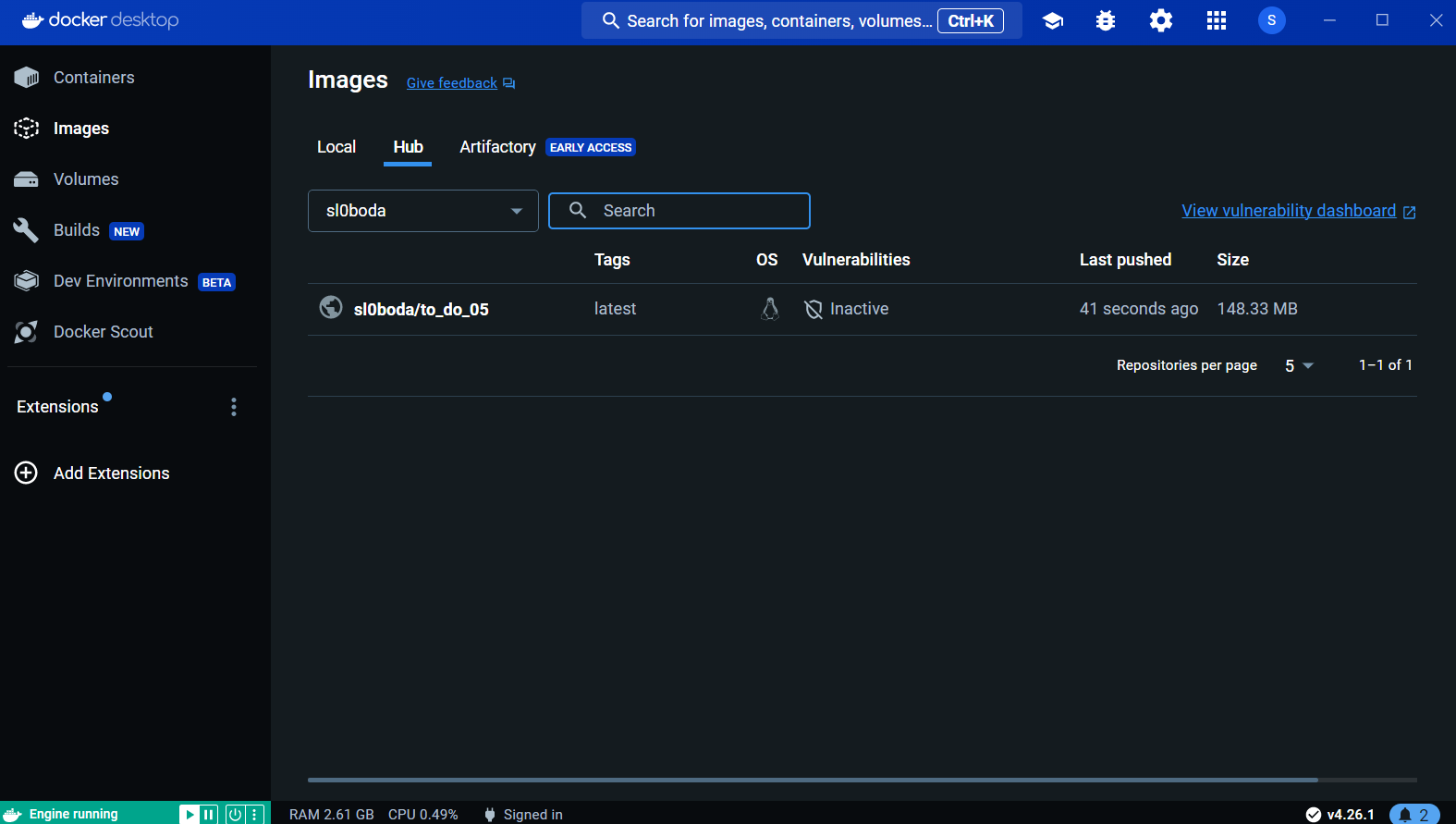
Part 2.

The command "docker buildx build --platform linux/amd64 --tag your-docker-hub-name/project-name" is used for creation of the Docker image with specified platform and tag:

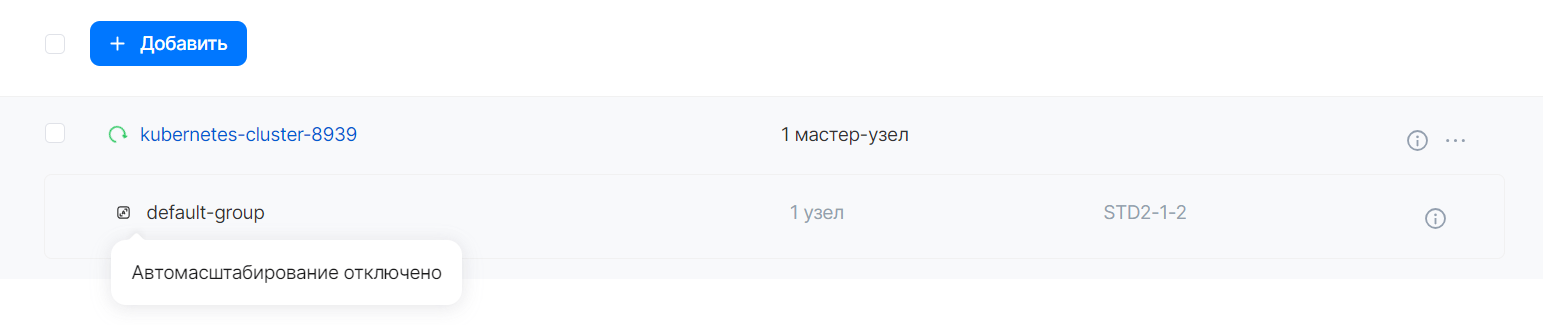


Pushing project using the DockerDesktop from local to the hub:

Now the image is on the hub.



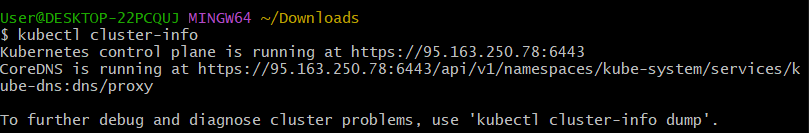
Let’s create and configure the K8s cluster.



Exporting the required variables and trying to connect to cluster:

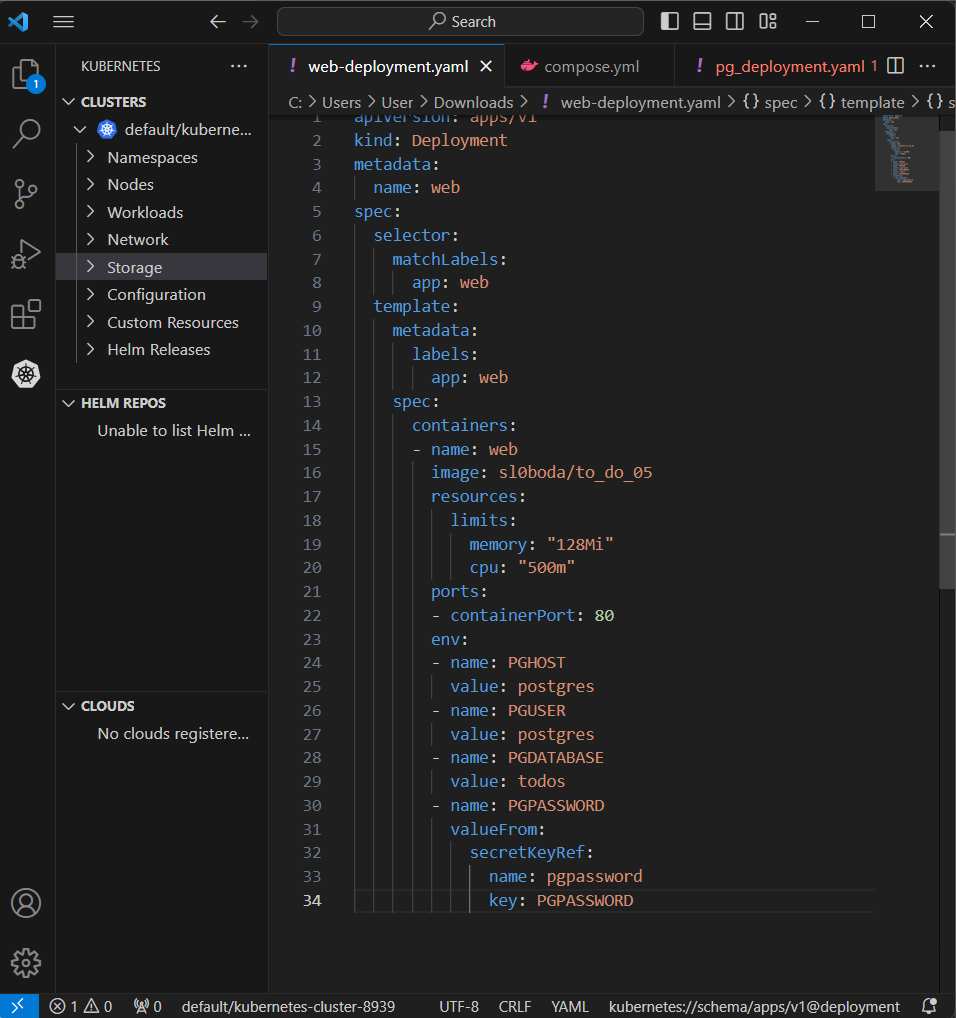
$ export KUBECONFIG=~/.kube/kubernetes-cluster-8939\_kubeconfig.yaml

$ source mcs4092792326-openrc.sh

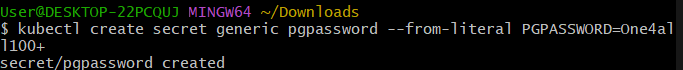


Successfully connected!

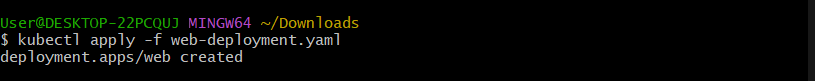
Next step is creation of the deployment .yaml file, here we will get the password from secret in privacy concerns:



Now let’s create the secret with pgpassword:

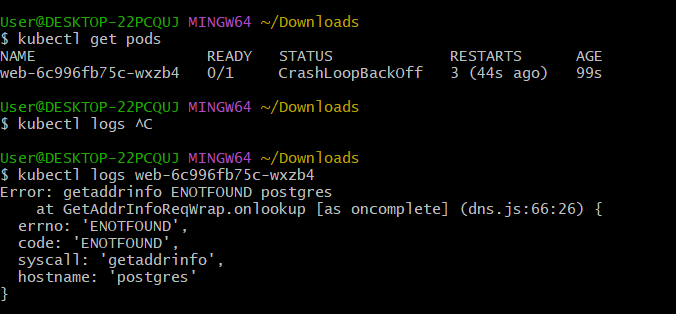


After that we can deploy the image:



And see that the pod was created:



The status is error, let’s see the logs:  


Obviously, it happens because we didn’t deploy the postgres yet.

Deployment file for postgres:

apiVersion: v1

kind: Service

metadata:

  name: postgres

spec:

  ports:

    - port: 5432

  selector:

    app: postgres

---

apiVersion: apps/v1

kind: Deployment

metadata:

  name: postgres

spec:

  selector:

    matchLabels:

      app: postgres

  template:

    metadata:

      labels:

        app: postgres

    spec:

      containers:

      - name: postgres

        image: postgres:14

        resources:

          limits:

            memory: "128Mi"

            cpu: "500m"

        env:

        - name: PGUSER

          value: postgres

        - name: PGPASSWORD

          valueFrom:

            secretKeyRef:

              name: pgpassword

              key: PGPASSWORD

        - name: PGDATABASE

          value: todo

        ports:

        - containerPort: 5432

        volumeMounts:

        - mountPath: /var/lib/postgresql/data

          name: postgredb

      volumes:

      - name: postgredb

        persistentVolumeClaim:

          claimName: postgres-pvc

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  name: postgres-pvc

spec:

  accessModes:

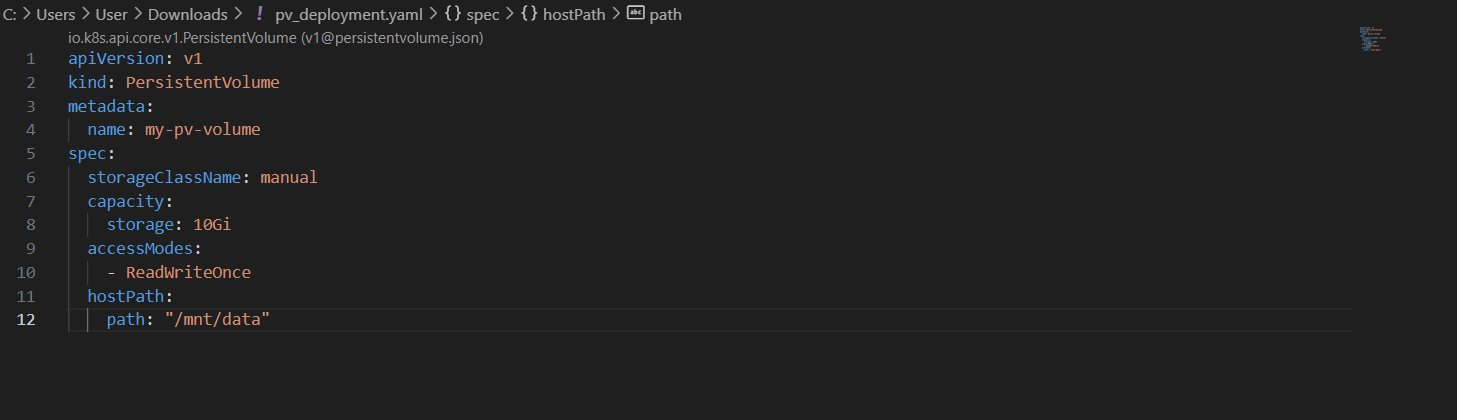
    - ReadWriteOnce

  resources:

    requests:

      storage: 1Gi

and deployment file for PV:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дальше на этапе создания PV не получилось сделать : (

Ошибка такая, что binding на PV не работает, потому что нужно указать хранилище, а какое указать – непонятно. Стандартного в списке не было, сил делать дальше тоже