بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

FATIMA JINNAH WOMEN UNIVERSITY,
RAWALPINDI

CLOUD COMPUTING
(BSE-410)

END SEMESTER PROJECT SUBMITTED TO
ENGR.WAQAS SALEEM

DEPARTMENT OF SOFTWARE ENGINEERING
SECTION B
BY
SYEDA FARWA BATOOL (2022-BSE-071)
EMAN ZAI (2022-BSE-049)

RAWALPINDI, PAKISTAN
DECEMBER 16, 2024

OBJECTIVE :

The objective of this project is to design and implement a Kubernetes-based micro-services architecture to gain practical experience in container orchestration. The project includes:

1. Creating at least three deployments (two for front-end micro-services and one for the back-end database).

2. Configuring Kubernetes services, including one external and two internal services.

3. Storing sensitive data using configuration maps and secrets.

4. Utilizing Docker containers and creating custom Docker images.

5. Documenting the implementation for reproducibility.

ARCHITECTURE:

The project architecture is designed as follows:

- **Front-end Deployments:**

  Two deployments represent the front-end services, each running in separate pods. These are responsible for handling user interactions.

- **Back-end Deployment:**

  A single deployment acts as the database backend, storing application data.

- **Services:**

  Each deployment is connected to a Kubernetes service:

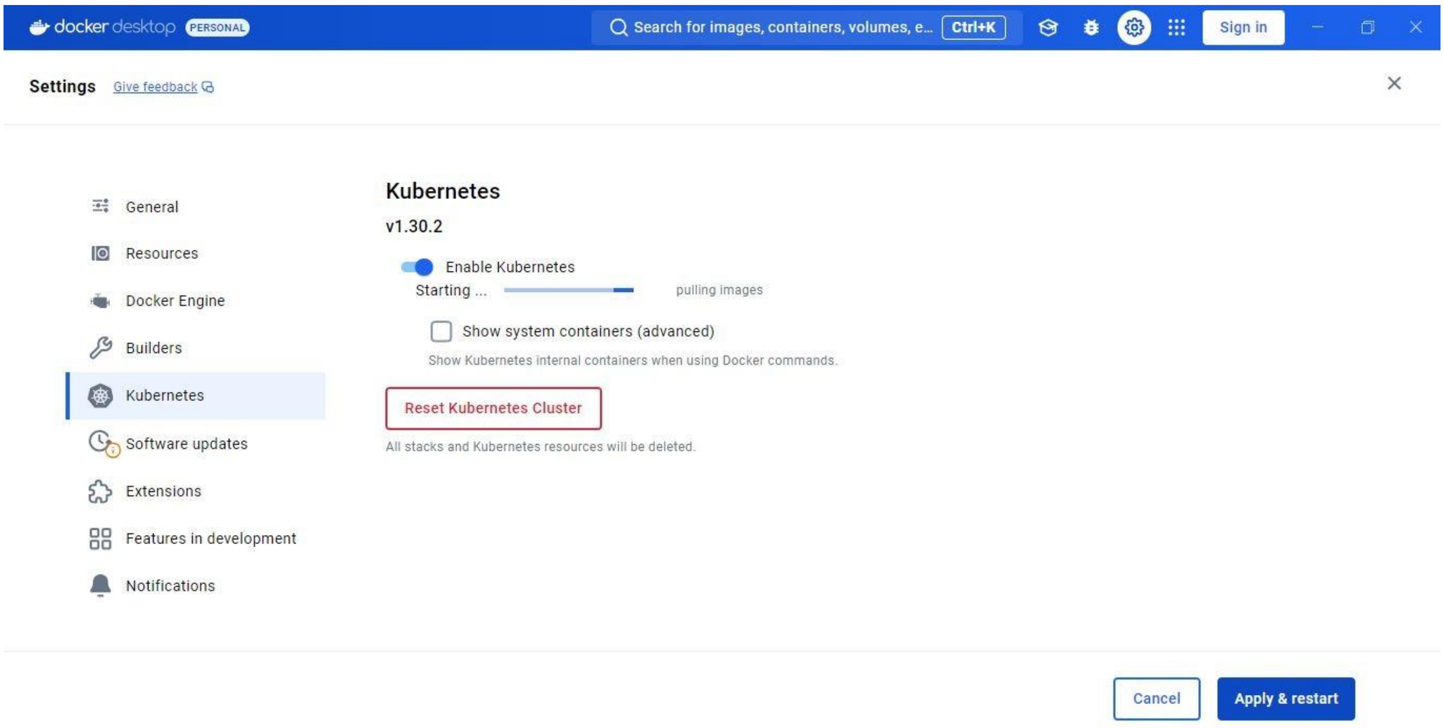
  - One external service allows access from outside the cluster.
  - Two internal services facilitate communication within the cluster.

- **Configuration Maps and Secrets:**

  - Configuration maps store non-sensitive configurations like database URLs.

Secrets store sensitive information such as database credentials.

Step 1: Creating Deployments



Kubernetes

v1.30.2

Enable Kubernetes

Starting ... pulling images

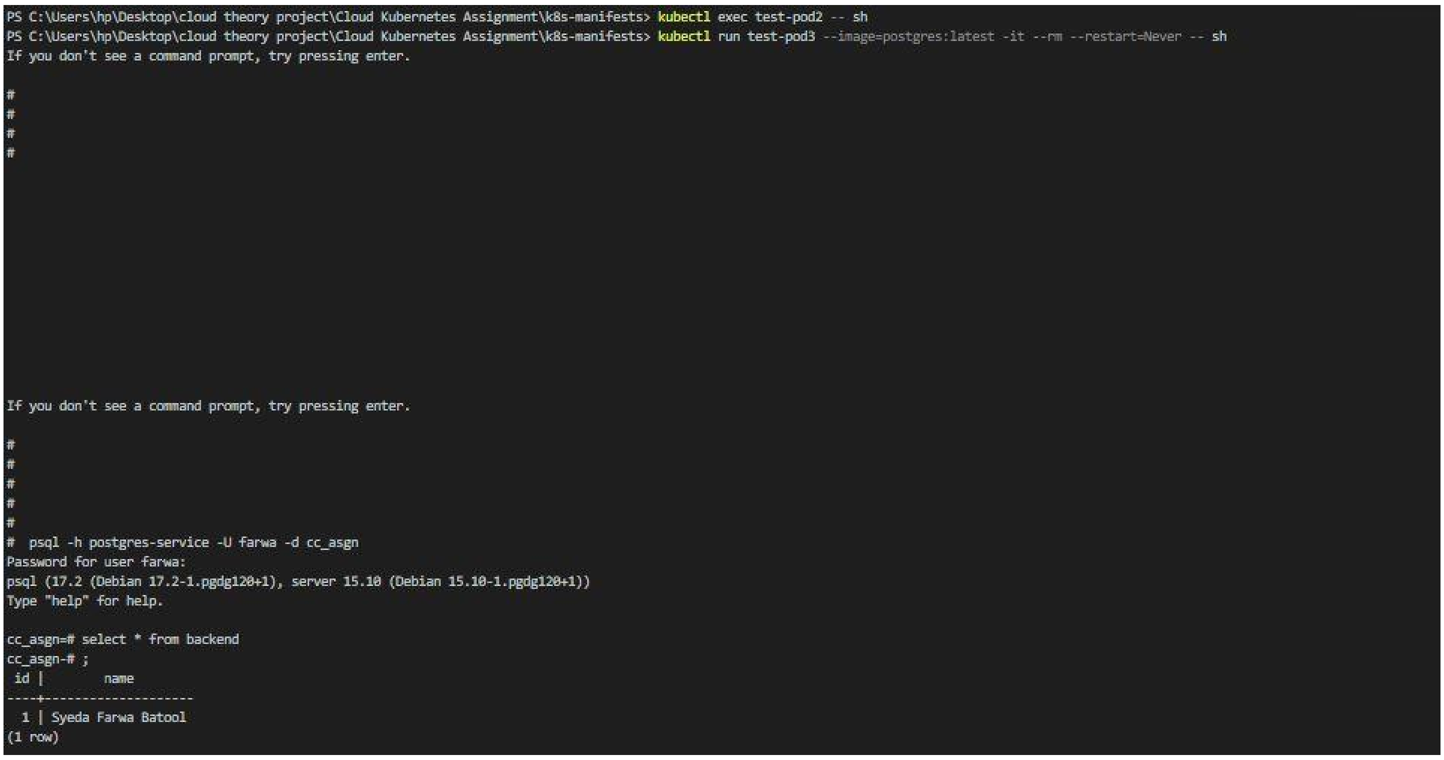
Show system containers (advanced)

Show Kubernetes internal containers when using Docker commands.

Reset Kubernetes Cluster

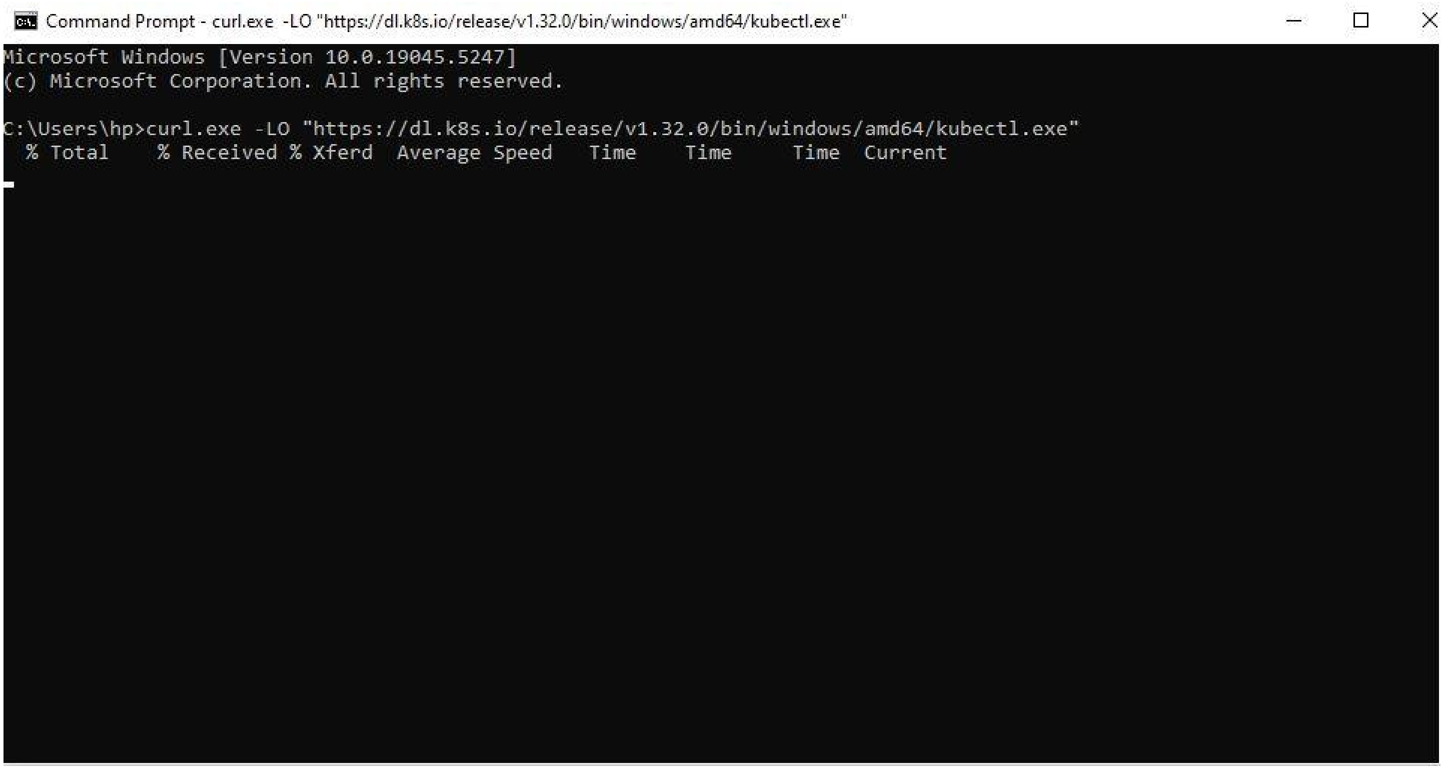
All stacks and Kubernetes resources will be deleted.

Step 2: Configuring Services



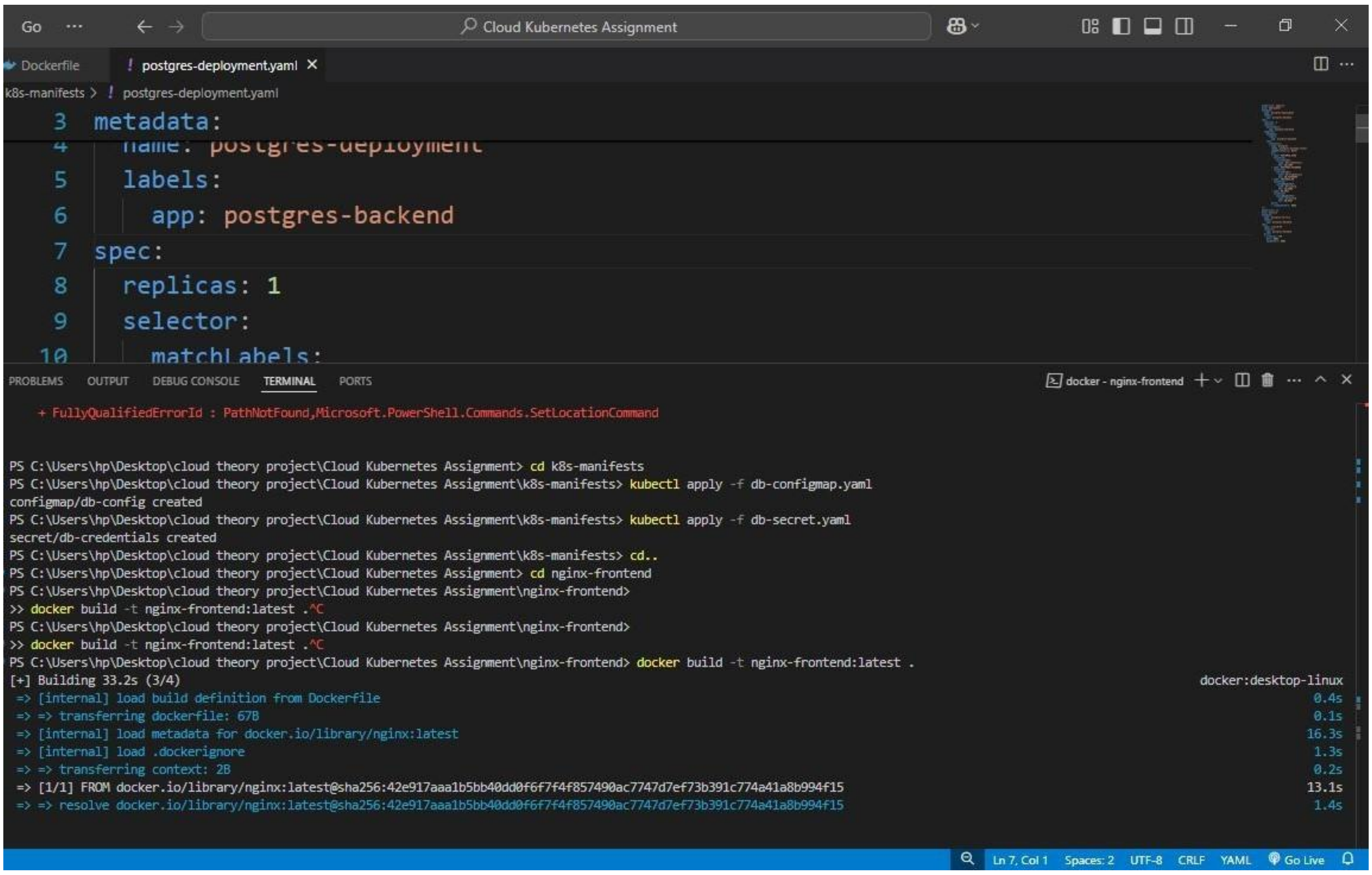
```
PS C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment\k8s-manifests> kubectl exec test-pod2 -- sh
PS C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment\k8s-manifests> kubectl run test-pod3 --image=postgres:latest -it --rm --restart=Never -- sh
If you don't see a command prompt, try pressing enter.

#
#
#
#









If you don't see a command prompt, try pressing enter.

#
#
#
#
#
#  psql -h postgres-service -U farwa -d cc_asgn
Password for user farwa:
psql (17.2 (Debian 17.2-1.pgdg120+1), server 15.10 (Debian 15.10-1.pgdg120+1))
Type "help" for help.

cc_asgn=# select * from backend
cc_asgn-# ;
 id |        name
----+--------------------
  1 | Syeda Farwa Batool
(1 row)
```

Step 3: Setting Up Configuration Maps and Secrets


Command Prompt - curl.exe -LO "https://dl.k8s.io/release/v1.32.0/bin/windows/amd64/kubectl.exe"

```
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>curl.exe -LO "https://dl.k8s.io/release/v1.32.0/bin/windows/amd64/kubectl.exe"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
```

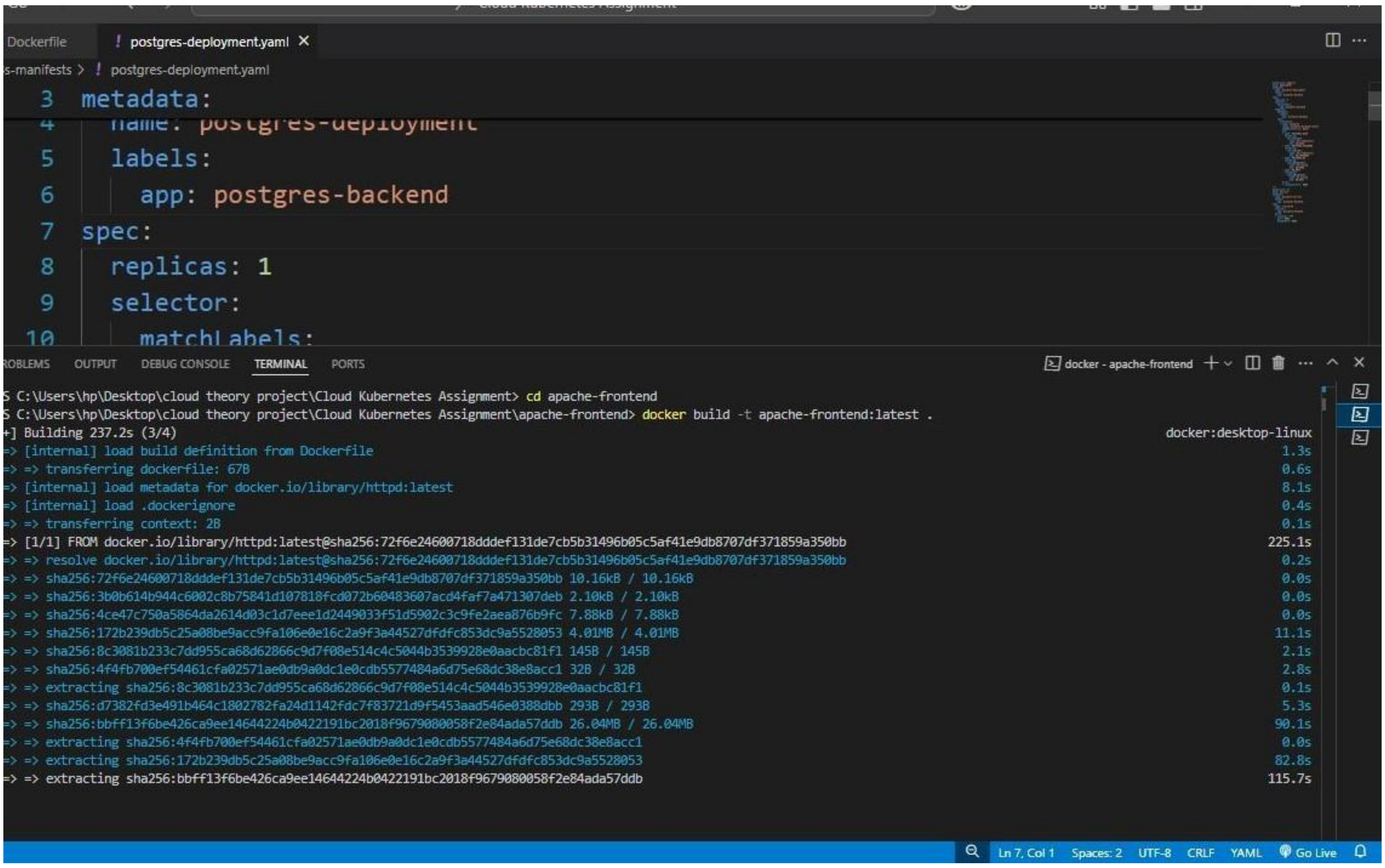
Step 4: Building Docker Images



```yaml
metadata:
  name: postgres-deployment
  labels:
    app: postgres-backend
spec:
  replicas: 1
  selector:
    matchLabels:
```

```
    + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocationCommand


PS C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment> cd k8s-manifests
PS C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment\k8s-manifests> kubectl apply -f db-configmap.yaml
configmap/db-config created
PS C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment\k8s-manifests> kubectl apply -f db-secret.yaml
secret/db-credentials created
PS C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment\k8s-manifests> cd..
PS C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment> cd nginx-frontend
PS C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment\nginx-frontend>
>> docker build -t nginx-frontend:latest .^C
PS C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment\nginx-frontend>
>> docker build -t nginx-frontend:latest .^C
PS C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment\nginx-frontend> docker build -t nginx-frontend:latest .
[+] Building 33.2s (3/4)                                                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                    0.4s
 => => transferring dockerfile: 67B                                                                                     0.1s
 => [internal] load metadata for docker.io/library/nginx:latest                                                        16.3s
 => [internal] load .dockerignore                                                                                       1.3s
 => => transferring context: 2B                                                                                         0.2s
 => [1/1] FROM docker.io/library/nginx:latest@sha256:42e917aaa1b5bb40dd0f6f7f4f857490ac7747d7ef73b391c774a41a8b994f15  13.1s
 => => resolve docker.io/library/nginx:latest@sha256:42e917aaa1b5bb40dd0f6f7f4f857490ac7747d7ef73b391c774a41a8b994f15   1.4s
```

Step 5: Deploying Resources on the Kubernetes Cluster

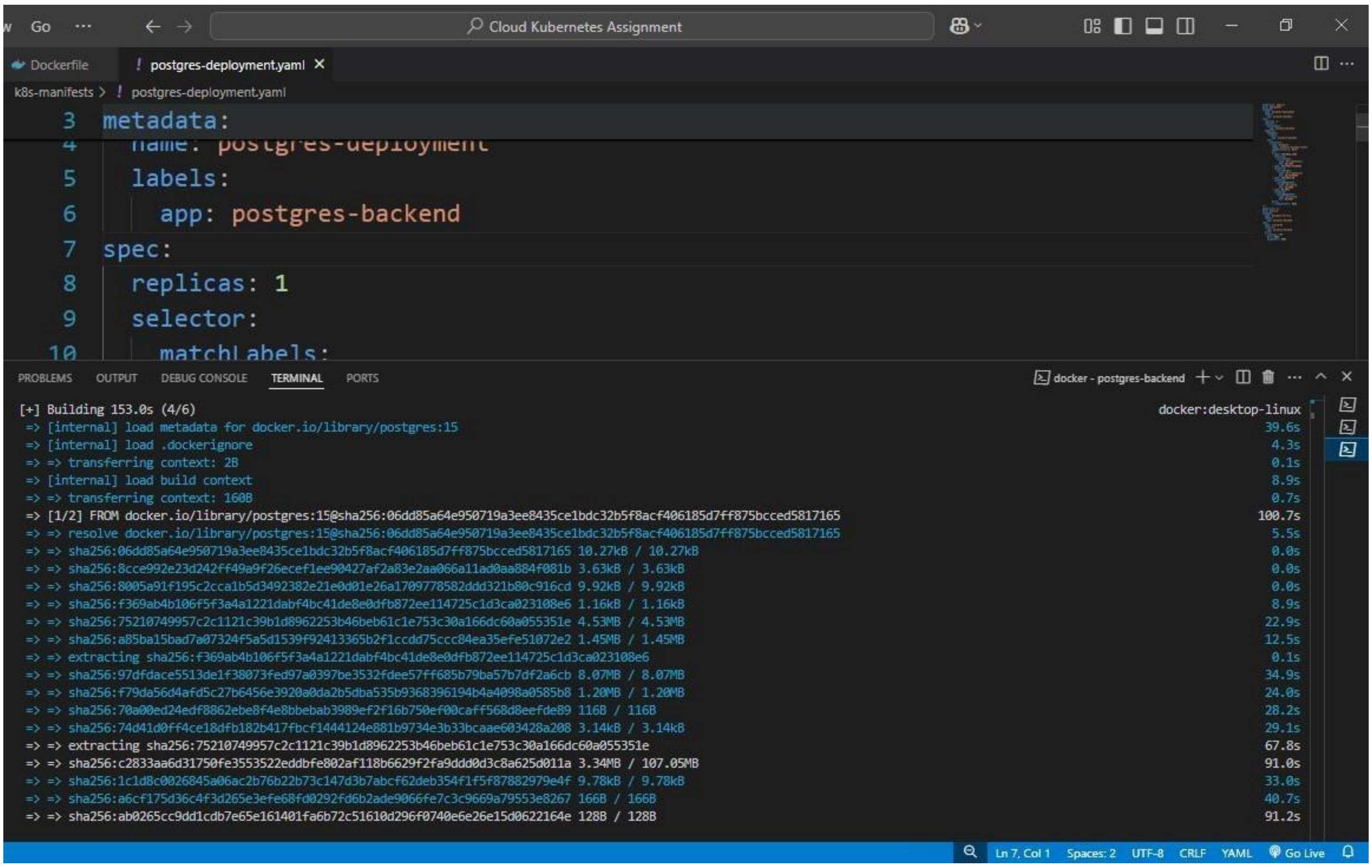


Dockerfile
postgres-deployment.yaml
s-manifests > postgres-deployment.yaml
ROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
docker - apache-frontend
Ln 7, Col 1 Spaces: 2 UTF-8 CRLF YAML Go Live

```yaml
metadata:
  name: postgres-deployment
  labels:
    app: postgres-backend
spec:
  replicas: 1
  selector:
    matchLabels:
```

```
S C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment> cd apache-frontend
S C:\Users\hp\Desktop\cloud theory project\Cloud Kubernetes Assignment\apache-frontend> docker build -t apache-frontend:latest .
+] Building 237.2s (3/4)                                                                          docker:desktop-linux
=> [internal] load build definition from Dockerfile                                                               1.3s
=> => transferring dockerfile: 67B                                                                                0.6s
=> [internal] load metadata for docker.io/library/httpd:latest                                                    8.1s
=> [internal] load .dockerignore                                                                                  0.4s
=> => transferring context: 2B                                                                                    0.1s
=> [1/1] FROM docker.io/library/httpd:latest@sha256:72f6e24600718dddef131de7cb5b31496b05c5af41e9db8707df371859a350bb  225.1s
=> => resolve docker.io/library/httpd:latest@sha256:72f6e24600718dddef131de7cb5b31496b05c5af41e9db8707df371859a350bb  0.2s
=> => sha256:72f6e24600718dddef131de7cb5b31496b05c5af41e9db8707df371859a350bb 10.16kB / 10.16kB                  0.0s
=> => sha256:3b0b614b944c6002c8b75841d107818fcd072b60483607acd4faf7a471307deb 2.10kB / 2.10kB                    0.0s
=> => sha256:4ce47c750a5864da2614d03c1d7eee1d2449033f51d5902c3c9fe2aea876b9fc 7.88kB / 7.88kB                    0.0s
=> => sha256:172b239db5c25a08be9acc9fa106e0e16c2a9f3a44527dfdfc853dc9a5528053 4.01MB / 4.01MB                   11.1s
=> => sha256:8c3081b233c7dd955ca68d62866c9d7f08e514c4c5044b3539928e0aacbc81f1 145B / 145B                        2.1s
=> => sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1 32B / 32B                          2.8s
=> => extracting sha256:8c3081b233c7dd955ca68d62866c9d7f08e514c4c5044b3539928e0aacbc81f1                        0.1s
=> => sha256:d7382fd3e491b464c1802782fa24d1142fdc7f83721d9f5453aad546e0388dbb 293B / 293B                        5.3s
=> => sha256:bbff13f6be426ca9ee14644224b0422191bc2018f9679080058f2e84ada57ddb 26.04MB / 26.04MB                 90.1s
=> => extracting sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1                        0.0s
=> => extracting sha256:172b239db5c25a08be9acc9fa106e0e16c2a9f3a44527dfdfc853dc9a5528053                       82.8s
=> => extracting sha256:bbff13f6be426ca9ee14644224b0422191bc2018f9679080058f2e84ada57ddb                      115.7s
```

Step 6: Accessing the External Service



Go
Cloud Kubernetes Assignment
Dockerfile
postgres-deployment.yaml
k8s-manifests > postgres-deployment.yaml
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
docker - postgres-backend
Ln 7, Col 1 Spaces: 2 UTF-8 CRLF YAML Go Live

```yaml
metadata:
  name: postgres-deployment
  labels:
    app: postgres-backend
spec:
  replicas: 1
  selector:
    matchLabels:
```

```
[+] Building 153.0s (4/6)                                                                         docker:desktop-linux
 => [internal] load metadata for docker.io/library/postgres:15                                                   39.6s
 => [internal] load .dockerignore                                                                                 4.3s
 => => transferring context: 2B                                                                                   0.1s
 => [internal] load build context                                                                                 8.9s
 => => transferring context: 160B                                                                                 0.7s
 => [1/2] FROM docker.io/library/postgres:15@sha256:06dd85a64e950719a3ee8435ce1bdc32b5f8acf406185d7ff875bcced5817165  100.7s
 => => resolve docker.io/library/postgres:15@sha256:06dd85a64e950719a3ee8435ce1bdc32b5f8acf406185d7ff875bcced5817165  5.5s
 => => sha256:06dd85a64e950719a3ee8435ce1bdc32b5f8acf406185d7ff875bcced5817165 10.27kB / 10.27kB                 0.0s
 => => sha256:8cce992e23d242ff49a9f26ecef1ee90427af2a83e2aa066a11ad0aa884f081b 3.63kB / 3.63kB                   0.0s
 => => sha256:8005a91f195c2cca1b5d3492382e21e0d01e26a1709778582ddd321b80c916cd 9.92kB / 9.92kB                   0.0s
 => => sha256:f369ab4b106f5f3a4a1221dabf4bc41de8e0dfb872ee114725c1d3ca023108e6 1.16kB / 1.16kB                   8.9s
 => => sha256:75210749957c2c1121c39b1d8962253b46beb61c1e753c30a166dc60a055351e 4.53MB / 4.53MB                  22.9s
 => => sha256:a85ba15bad7a07324f5a5d1539f92413365b2f1ccdd75ccc84ea35efe51072e2 1.45MB / 1.45MB                  12.5s
 => => extracting sha256:f369ab4b106f5f3a4a1221dabf4bc41de8e0dfb872ee114725c1d3ca023108e6                       0.1s
 => => sha256:97dfdace5513de1f38073fed97a0397be3532fdee57ff685b79ba57b7df2a6cb 8.07MB / 8.07MB                  34.9s
 => => sha256:f79da56d4afd5c27b6456e3920a0da2b5dba535b9368396194b4a4098a0585b8 1.20MB / 1.20MB                  24.0s
 => => sha256:70a00ed24edf8862ebe8f4e8bbebab3989ef2f16b750ef00caff568d8eefde89 116B / 116B                      28.2s
 => => sha256:74d41d0ff4ce18dfb182b417fbcf1444124e881b9734e3b33bcaae603428a208 3.14kB / 3.14kB                  29.1s
 => => extracting sha256:75210749957c2c1121c39b1d8962253b46beb61c1e753c30a166dc60a055351e                      67.8s
 => => sha256:c2833aa6d31750fe3553522eddbfe802af118b6629f2fa9ddd0d3c8a625d011a 3.34MB / 107.05MB                91.0s
 => => sha256:1c1d8c0026845a06ac2b76b22b73c147d3b7abcf62deb354f1f5f87882979e4f 9.78kB / 9.78kB                  33.0s
 => => sha256:a6cf175d36c4f3d265e3efe68fd0292fd6b2ade9066fe7c3c9669a79553e8267 166B / 166B                      40.7s
 => => sha256:ab0265cc9dd1cdb7e65e161401fa6b72c51610d296f0740e6e26e15d0622164e 128B / 128B                      91.2s
```

Challenges and Solutions:

During the project, the following challenges were faced:

1. Configuring Kubernetes YAML manifests.
2. Setting up secrets securely.
3. Debugging issues in service connectivity.

Conclusion:

This project provided hands-on experience with Kubernetes and Docker. The architecture demonstrated a scalable micro-services model, and the implementation taught critical skills in container orchestration, service management, and security practices. The project successfully met all objectives, and potential improvements include automating deployments using CI/CD pipelines