Scalable Microservices Architecture using Docker + Kubernetes + NodePort

Microservices:

- **User Service** (/users)
- **Product Service** (/products)
- **Order Service** (/orders)

Step 1: Development for each microservice (Setup Dockerfile and pages of each service)

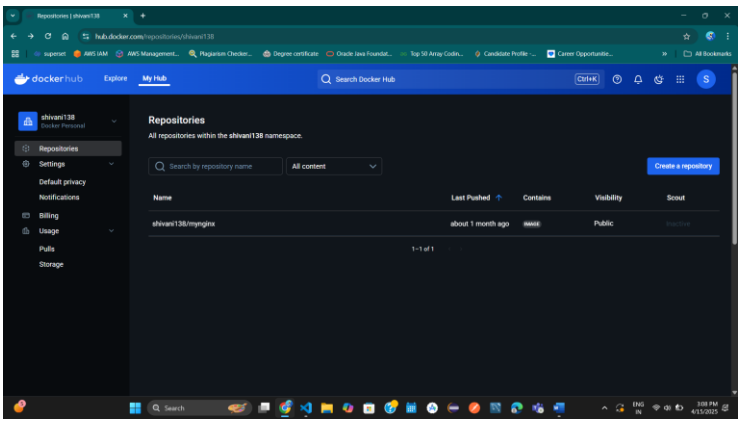
Common structure (for all 3 services)

Each folder will have:

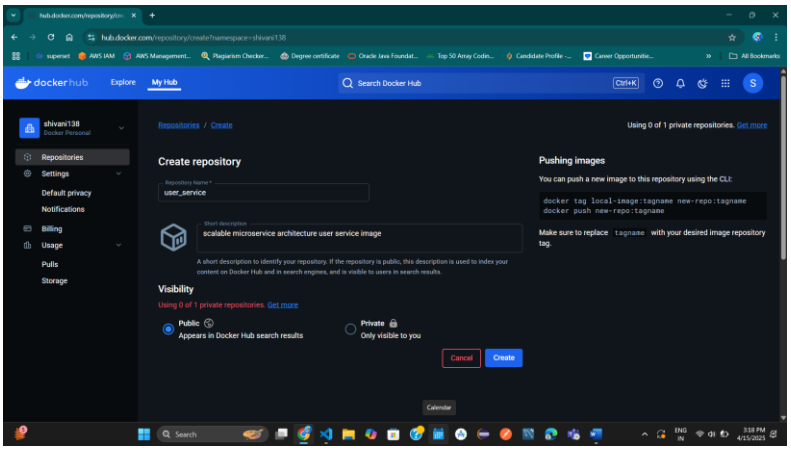
- app.py (or JS file if using Node)
- requirements.txt (for Flask)
- Dockerfile

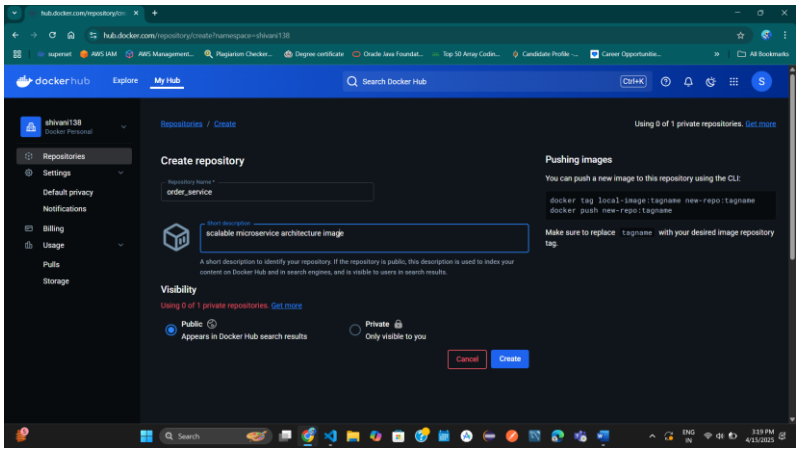
Step 2: Build and Push Docker Images (create image from dockerfile and push images to dockerhub)

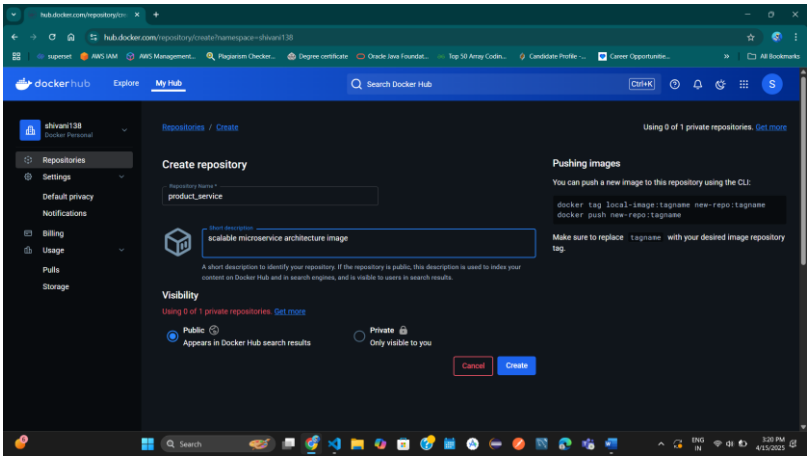
- **Login to docker hub account on browser**



- **Create repo to store image**







- **Generate personal access token**

  Dockerhub on browser → profile → account settings → personal access token → generate.

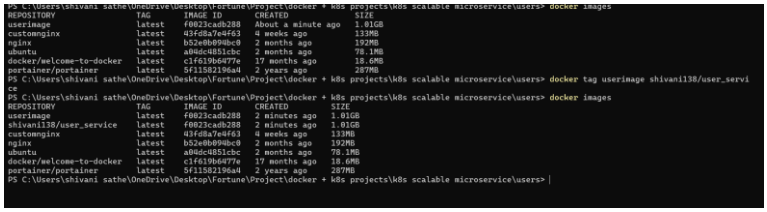
- **Login to dockerhub account on desktop powershell**

  docker login – provide dockerhub username and token as password

- **Run these commands for each microservice: (Repeat for 3 services)**
  1) Create image from dockerfile
     docker build -t userimage .

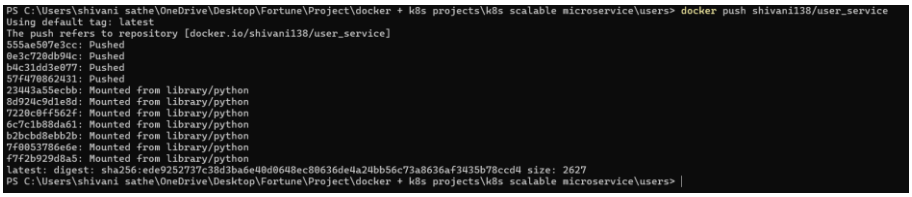
2) Tag image

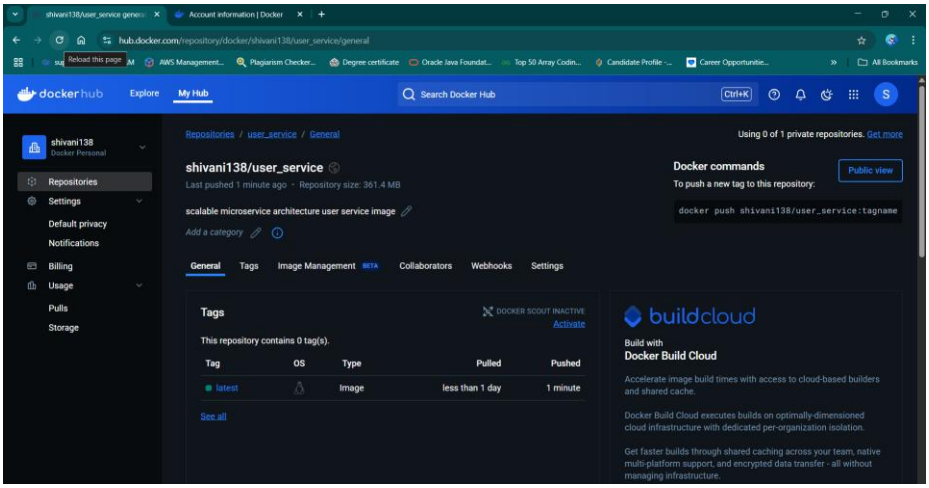
Docker tag image_name username/repo_name



3) Push image to dockerhub

docker push username/repo_name





Step 3: Create Kubernetes Deployment and Service Files

user-deployment.yaml

product-deployment.yaml

order-deployment.yaml

Note – code added in repo and create files on master node.

Step 4: Apply YAMLs

kubectl apply -f user-deployment.yaml

kubectl apply -f product-deployment.yaml

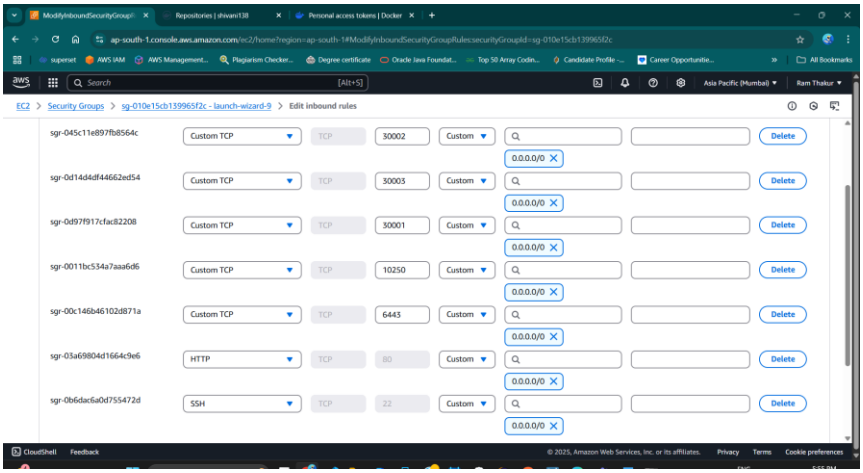
kubectl apply -f order-deployment.yaml

```
ubuntu@ip-172-31-3-220: ~
ubuntu@ip-172-31-3-220:~$ kubectl apply -f order-deployment.yaml
deployment.apps/order-service created
service/order-service created
ubuntu@ip-172-31-3-220:~$ kubectl apply -f product-deployment.yaml
deployment.apps/product-service created
service/product-service created
ubuntu@ip-172-31-3-220:~$ kubectl apply -f user_deployment.yaml
deployment.apps/user-service created
service/user-service created
```

```
ubuntu@ip-172-31-3-220:~$ kubectl get all
NAME                                   READY   STATUS    RESTARTS   AGE
pod/order-service-7885ff7d5c-cd8cd     1/1     Running   0          65s
pod/product-service-c788cd55c-c5cgb    1/1     Running   0          19s
pod/user-service-f7d4c6b88-bk2pz       1/1     Running   0          4m

NAME                      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/kubernetes        ClusterIP   10.96.0.1        <none>        443/TCP        4h50m
service/order-service     NodePort    10.109.248.194   <none>        80:30003/TCP   65s
service/product-service   NodePort    10.109.198.175   <none>        80:30002/TCP   19s
service/user-service      NodePort    10.105.45.58     <none>        80:30001/TCP   4m

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/order-service     1/1     1            1           65s
deployment.apps/product-service   1/1     1            1           19s
deployment.apps/user-service      1/1     1            1           4m

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/order-service-7885ff7d5c     1         1         1       65s
replicaset.apps/product-service-c788cd55c    1         1         1       19s
replicaset.apps/user-service-f7d4c6b88       1         1         1       4m
ubuntu@ip-172-31-3-220:~$
```

Step 5: Test Microservices

Note - Add port 30001, 30002, 30003 in worker node sg



Go to browser and search

http://localhost:30001/users

http://localhost:30002/products

http://localhost:30003/orders

If using EC2 instance replace localhost with worker node ip