

Assignment 3:

1. Dockerized the Node.js (Express) App:

The screenshot shows a terminal window titled "ashraful [SSH: VM2]". The window contains a code editor with several tabs: "2. command_cheap_sheet.md", "1. important_command.md", "3. fix docker command.md", "app.js", "Dockerfile" (which is the active tab), "docker-compose.yml", "default.conf", "my.cnf", "package-lock.json", and "1.". The "Dockerfile" tab displays the following content:

```
DigitalCardServiceApp > Dockerfile
1  FROM node:20-slim
2
3  # Set working directory
4  WORKDIR /app
5
6  # Copy package files
7  COPY package*.json ./
8
9  # Install dependencies
10 RUN npm install --production
11
12 # Copy application code
13 COPY .
14
15 # Expose port
16 EXPOSE 3000
17
18 # Health check
19 HEALTHCHECK --interval=30s --timeout=3s --start-period=40s --retries=3 \
20 | CMD node -e "require('http').get('http://localhost:3000/health', (r) => {if (r.statusCode != 200) throw new Error(r.statusCode)})"
21
22 # Start application
23 CMD ["node", "app.js"]
24 # CMD ["npx", "nodemon", "app.js"]
25
```

The status bar at the bottom indicates "SSH: VM2" and "master*".

Build Image from Dockerfile & Tag :

```
>> docker build -t express-app:1.0 .
>> docker tag express-app:1.2 ashrafulinstasure/digital_card:1.3
>> docker push ashrafulinstasure/digital_card:1.2
```

#Docker image on the Docker Hub:

The screenshot shows the Docker Hub interface for the repository `ashrafulinstasure/digital_card`. The repository was last pushed 1 day ago and has a size of 72.6 MB. It contains two tags: `1.3` and `1.2`. The `General` tab is selected. A sidebar on the left provides navigation links for repositories, hardened images, collaborations, settings, billing, usage, pulls, and storage. A right-hand sidebar displays an advertisement for buildcloud.

Docker commands
To push a new tag to this repository:
`docker push ashrafulinstasure/digital_card`

Tag	OS	Type	Pulled	Pushed
1.3	Ubuntu	Image	1 day	1 day
1.2	Ubuntu	Image	2 days	2 days

Repository overview INCOMPLETE

An overview describes what your image does and how to run it. It displays in [the public view of your repository](#) once you have pushed some content.

[Add overview](#)

buildcloud
Build with Docker Build Cloud
Accelerate image build times with access to cloud-based infrastructure.
Docker Build Cloud executes builds on optimally-dimensioned, dedicated per-organization isolation.
Get faster builds through shared caching across your organization and encrypted data transfer - all without managing infrastructure.

[Go to Docker Build Cloud →](#)

2. Docker-Compose file that run Nginx, Mysql, redis, rabbitMQ with Bridge Network and Volume Persistance .

```

version: '3.8'

services:
  # Node.js App with Express
  node-app:
    image: ashrafulinstasure/digital_card:1.3
    container_name: node-app
    restart: on-failure
    working_dir: /var/www/html
    environment:
      - NODE_ENV=production
      - PORT=3000
      - DB_HOST=mysql-node
      - DB_PORT=3306
      - DB_DATABASE=nodejs_db
      - DB_USERNAME=nodejs_user
      - DB_PASSWORD=nodejs_password
      - REDIS_HOST=redis-node
      - REDIS_PORT=6379
      - RABBITMQ_HOST=rabbitmq-node
      - RABBITMQ_PORT=5672
      - RABBITMQ_USER=guest
      - RABBITMQ_PASSWORD=guest
      - RABBITMQ_VHOST=/guest
      - APP_URL=http://36.255.69.72:8080
    volumes:
      - ./var/www/html
    ports:
      - 3000:3000
    networks:
      - nodejs_network
    depends_on:
      - mysql-node
      - redis-node
      - rabbitmq-node
  # Nginx for Node.js
  nginx-node:
    image: nginx:alpine
    container_name: nginx-node
    restart: unless-stopped
    ports:
      - 8080:80
      - 8443:443
    volumes:
      - ./var/www/html
      - ./docker/nginx/default.conf:/etc/nginx/conf.d/default.conf
    networks:
      - nodejs_network
    depends_on:
      - node-app

```

3. docker-compose.yml :

```

version: '3.8'

services:
  # Node.js App with Express
  node-app:
    image: ashrafulinstasure/digital_card:1.3
    container_name: node-app
    restart: on-failure
    working_dir: /var/www/html
    environment:
      - NODE_ENV=production
      - PORT=3000
      - DB_HOST=mysql-node
      - DB_PORT=3306
      - DB_DATABASE=nodejs_db
      - DB_USERNAME=nodejs_user
      - DB_PASSWORD=nodejs_password
      - REDIS_HOST=redis-node
      - REDIS_PORT=6379
      - RABBITMQ_HOST=rabbitmq-node
      - RABBITMQ_PORT=5672

```

```
- RABBITMQ_USER=guest
- RABBITMQ_PASSWORD=guest
- RABBITMQ_VHOST=/
- APP_URL=http://36.255.69.72:8080

volumes:
- ./var/www/html

ports:
- "3000:3000"

networks:
- nodejs_network

depends_on:
- mysql-node
- redis-node
- rabbitmq-node

# Nginx for Node.js

nginx-node:
image: nginx:alpine
container_name: nginx-node
restart: unless-stopped
ports:
- "8080:80"
- "8443:443"
volumes:
- ./var/www/html
- ./docker/nginx/default.conf:/etc/nginx/conf.d/default.conf
networks:
- nodejs_network
depends_on:
- node-app

# MySQL for Node.js

mysql-node:
image: mysql:8.0
container_name: mysql-node
restart: unless-stopped
environment:
MYSQL_DATABASE: nodejs_db
MYSQL_ROOT_PASSWORD: nodejs_root_password
MYSQL_USER: nodejs_user
```

```
    MYSQL_PASSWORD: nodejs_password
  ports:
    - "3308:3306"
  volumes:
    - mysql_node_data:/var/lib/mysql
    - ./docker/mysql/my.cnf:/etc/mysql/my.cnf
  networks:
    - nodejs_network
  healthcheck:
    test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
    interval: 10s
    timeout: 5s
    retries: 5

# Redis for Node.js
redis-node:
  image: redis:7-alpine
  container_name: redis-node
  restart: unless-stopped
  ports:
    - "6381:6379"
  volumes:
    - redis_node_data:/data
  networks:
    - nodejs_network
  healthcheck:
    test: ["CMD", "redis-cli", "ping"]
    interval: 10s
    timeout: 5s
    retries: 5

# RabbitMQ - Message Queue
rabbitmq-node:
  image: rabbitmq:3.12-management-alpine
  container_name: rabbitmq-node
  restart: unless-stopped
  environment:
    RABBITMQ_DEFAULT_USER: guest
    RABBITMQ_DEFAULT_PASS: guest
  ports:
```

```
- "5672:5672"
- "15672:15672"

volumes:
- rabbitmq_node_data:/var/lib/rabbitmq

networks:
- nodejs_network

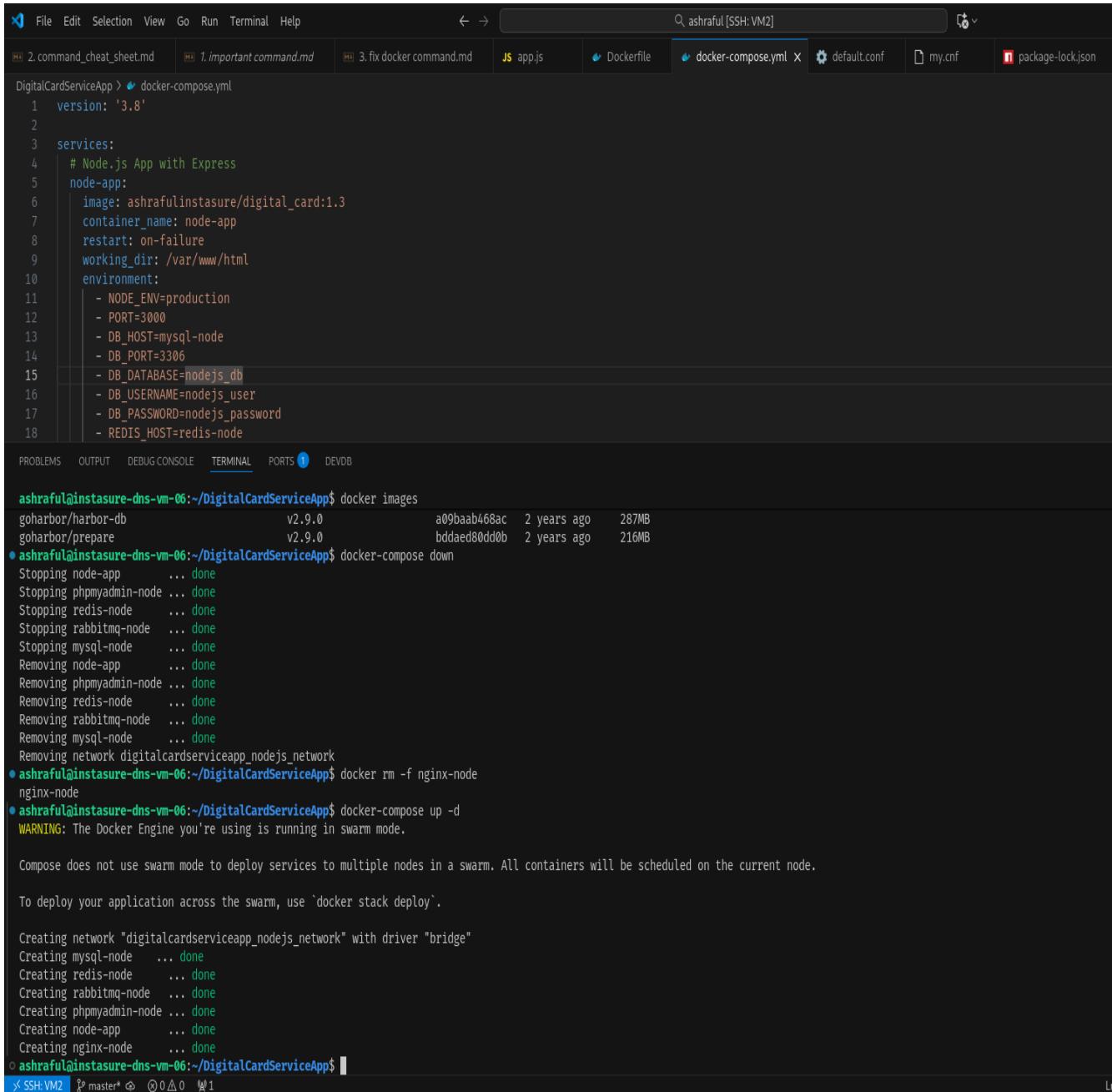
healthcheck:
  test: rabbitmq-diagnostics -q ping
  interval: 30s
  timeout: 10s
  retries: 5

# PHPMyAdmin for Node.js MySQL
phpmyadmin-node:
  image: phpmyadmin/phpmyadmin
  container_name: phpmyadmin-node
  restart: unless-stopped
  environment:
    PMA_HOST: mysql-node
    PMA_PORT: 3306
    PMA_USER: root
    PMA_PASSWORD: nodejs_root_password
  ports:
- "8082:80"
  networks:
- nodejs_network
  depends_on:
- mysql-node

volumes:
mysql_node_data:
  driver: local
redis_node_data:
  driver: local
rabbitmq_node_data:
  driver: local

networks:
nodejs_network:
  driver: bridge
```

4. Run the docker-compose.yml file :



The screenshot shows a terminal window with several tabs at the top: "File Edit Selection View Go Run Terminal Help", "ashraful [SSH:VM2]", "2. command_cheat_sheet.md", "1. important command.md", "3. fix docker command.md", "JS app.js", "Dockerfile", "docker-compose.yml", "default.conf", "my.cnf", and "package-lock.json". The "docker-compose.yml" tab is active.

The terminal content is as follows:

```
DigitalCardServiceApp > docker-compose.yml
1 version: '3.8'
2
3 services:
4   # Node.js App with Express
5   node-app:
6     image: ashrafulinstasure/digital_card:1.3
7     container_name: node-app
8     restart: on-failure
9     working_dir: /var/www/html
10    environment:
11      - NODE_ENV=production
12      - PORT=3000
13      - DB_HOST=mysql-node
14      - DB_PORT=3306
15      - DB_DATABASE=nodejs_db
16      - DB_USERNAME=nodejs_user
17      - DB_PASSWORD=nodejs_password
18      - REDIS_HOST=redis-node

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 DEVDDB

ashraful@instasure-dns-vm-06:~/DigitalCardServiceApp$ docker images
goharbor/harbor-db          v2.9.0           a09baab468ac  2 years ago  287MB
goharbor/prepare              v2.9.0           bddaed80dd0b  2 years ago  216MB
● ashraful@instasure-dns-vm-06:~/DigitalCardServiceApp$ docker-compose down
Stopping node-app ... done
Stopping phpmyadmin-node ... done
Stopping redis-node ... done
Stopping rabbitmq-node ... done
Stopping mysql-node ... done
Removing node-app ... done
Removing phpmyadmin-node ... done
Removing redis-node ... done
Removing rabbitmq-node ... done
Removing mysql-node ... done
Removing network digitalcardserviceapp_nodejs_network
● ashraful@instasure-dns-vm-06:~/DigitalCardServiceApp$ docker rm -f nginx-node
nginx-node
● ashraful@instasure-dns-vm-06:~/DigitalCardServiceApp$ docker-compose up
WARNING: The Docker Engine you're using is running in swarm mode.

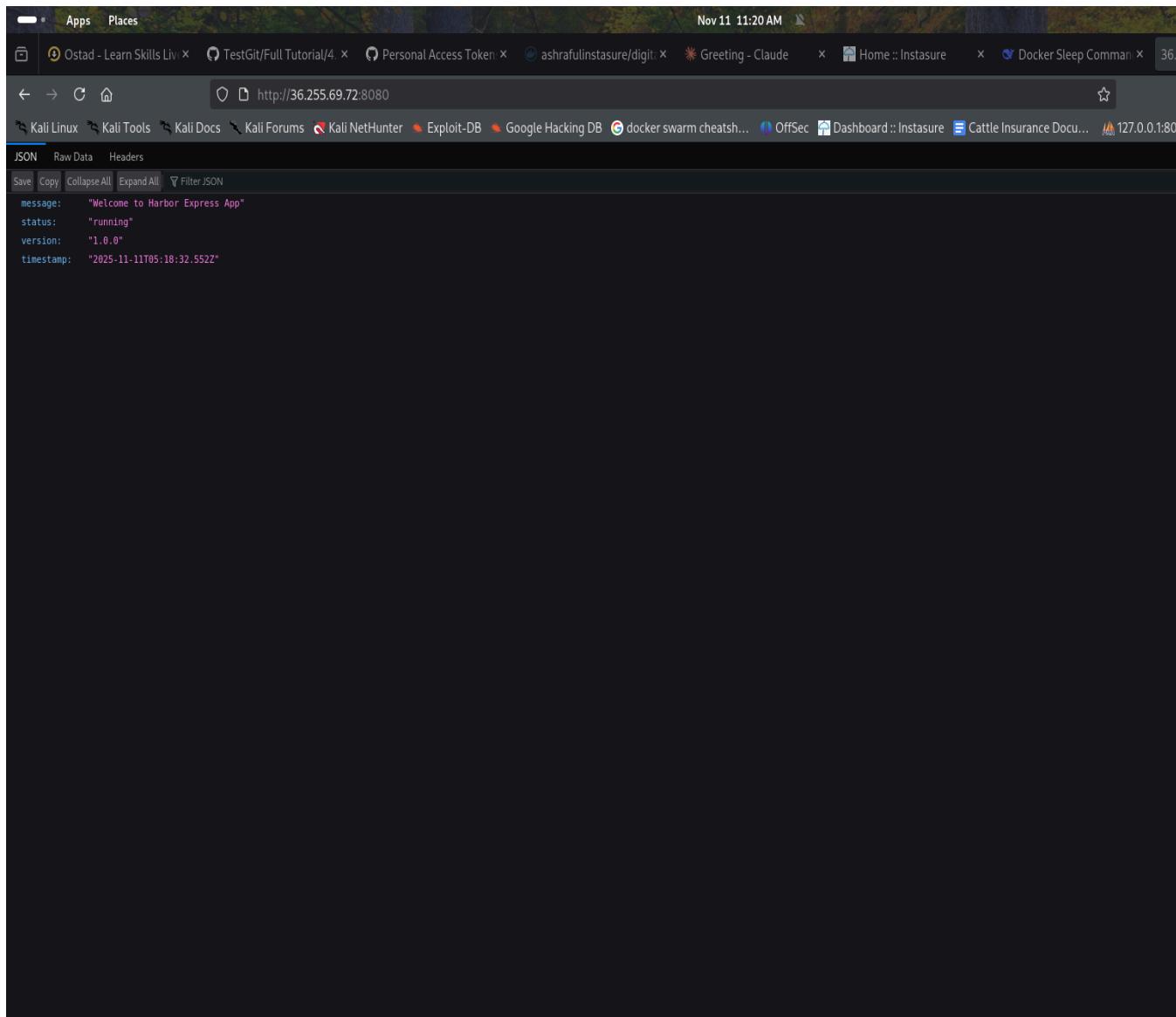
Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Creating network "digitalcardserviceapp_nodejs_network" with driver "bridge"
Creating mysql-node ... done
Creating redis-node ... done
Creating rabbitmq-node ... done
Creating phpmyadmin-node ... done
Creating node-app ... done
Creating nginx-node ... done
● ashraful@instasure-dns-vm-06:~/DigitalCardServiceApp$
```

The bottom status bar shows "SSH: VM2" and "master*".

5. Express On Runnig on the VPS -> http://36.255.69.72:8080/:



A screenshot of a terminal window titled "Kali Linux" showing the output of an Express application running on port 8080. The terminal has tabs for "Ostad - Learn Skills Liv...", "TestGit/FullTutorial/4...", "Personal Access Token", "ashrafulinstasure/digit...", "Greeting - Claude", "Home :: Instasure", "Docker Sleep Command", "Kali Tools", "Kali Docs", "Kali Forums", "Kali NetHunter", "Exploit-DB", "Google Hacking DB", "docker swarm cheatsheet", "OffSec", "Dashboard :: Instasure", "Cattle Insurance Docu...", and "127.0.0.1:80". The main pane displays the following JSON response:

```
message: "Welcome to Harbor Express App"
status: "running"
version: "1.0.0"
timestamp: "2025-11-11T05:18:32.552Z"
```

Below the JSON output, there are buttons for "Save", "Copy", "Collapse All", "Expand All", and "Filter JSON".