Docker Composed
Setting up a composed webserver

Introduction:

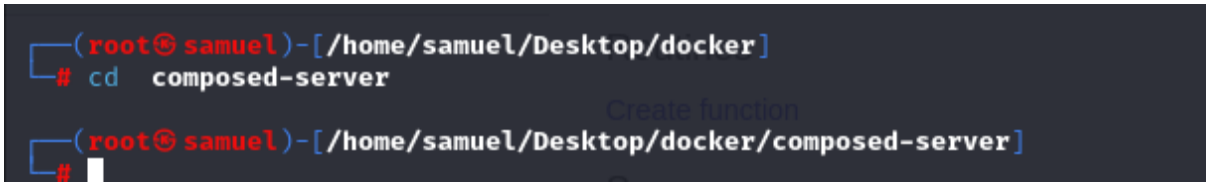
This report outlines the methodology for establishing a Docker Compose environment tailored for web application development. This environment will encompass three core components: a web server, a database, and an administrative interface. The deployment will leverage Docker and Docker Compose, ensuring seamless communication and data persistence among the services. A solid understanding of YAML syntax and web server configurations is a prerequisite for this endeavor.

Pre-requisites:

- Docker and Docker Compose are installed on your machine. Verify installation by running docker --version and docker-compose --version in your terminal.
- Basic understanding of YAML syntax and web server configurations.

Task 1: Docker Compose File Creation

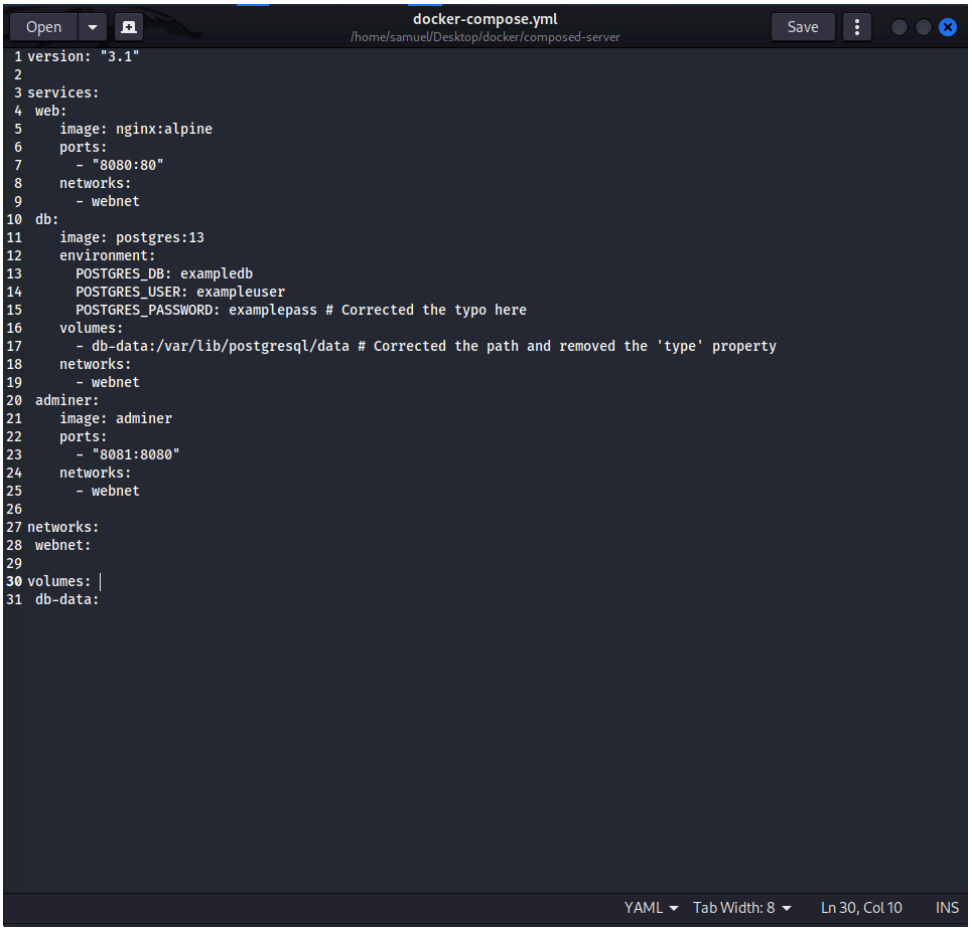
1. Create a directory for your project:
    - Create a new directory named composed-server.
    - Navigate into the directory.
    - 

    - 

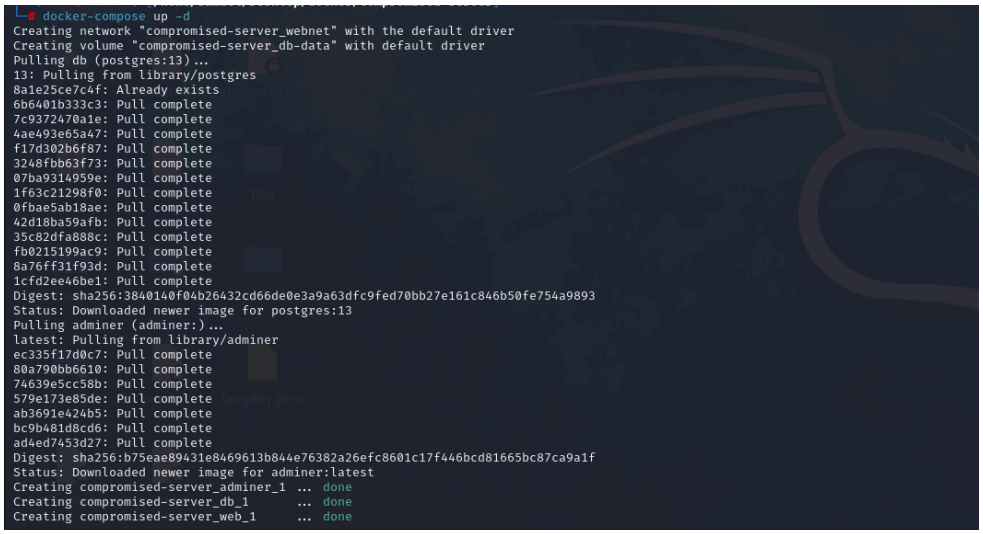
2. Create a Docker Compose file:
    - Inside the composed-server directory, create a file named docker-compose.yml.
    - Open the file in a text editor.

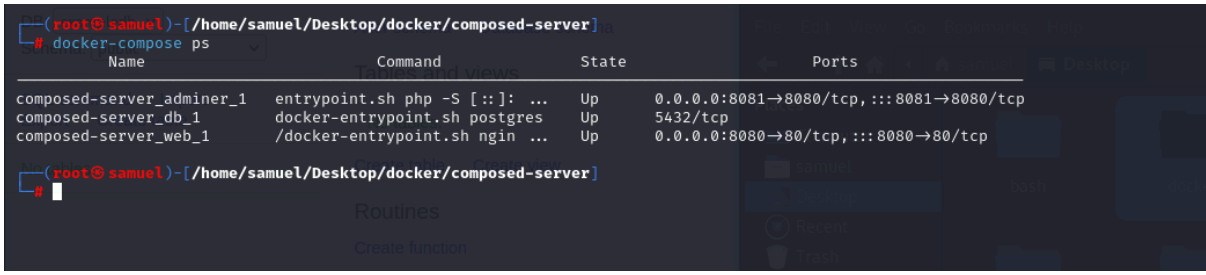
- 

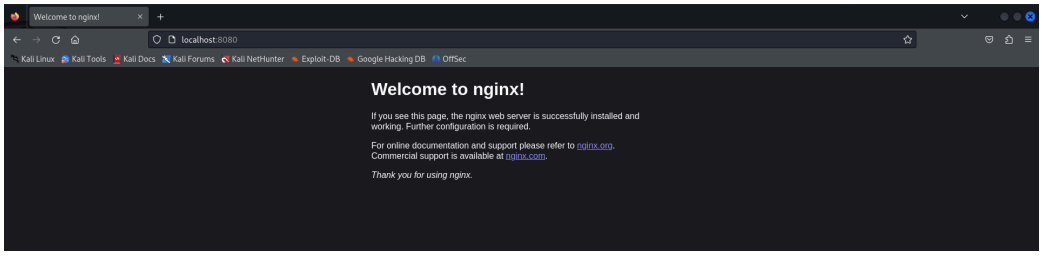
3. Define Services:
    - In the docker-compose.yml file, define three services: web, db, and adminer.
    - Use the nginx:alpine image for the web service.
    - Use the postgres:13 image for the db service. Define environment variables for POSTGRES_DB, POSTGRES_USER, and POSTGRES_PASSWORD.
    - Use the adminer image for the adminer service.
4. Configure Networking:
    - Create a custom network called webnet for your services to communicate.
    - Assign all three services to this network.
5. Volumes and Ports:
    - Mount a volume for the PostgreSQL data to persist data outside the container lifecycle.
    - Map port 8080 to the web service's port 80.
    - Map port 8081 to the adminer service's port 8080.



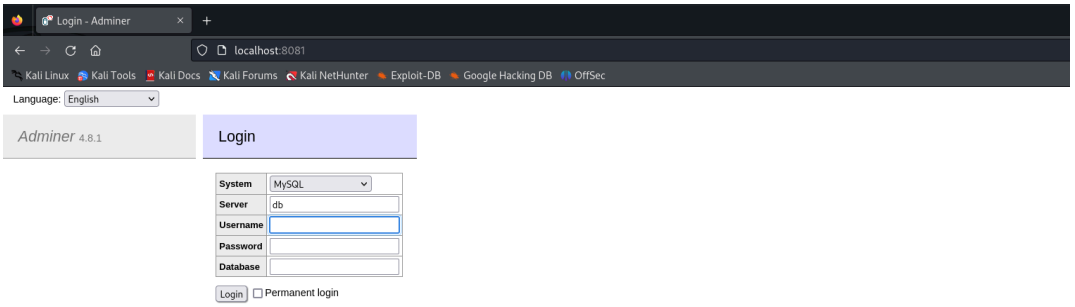
Task 2: Running and Testing Your Composed Environment

1. Starting Services:
    - Run your composed environment with the following command: docker-compose up -d
    - 

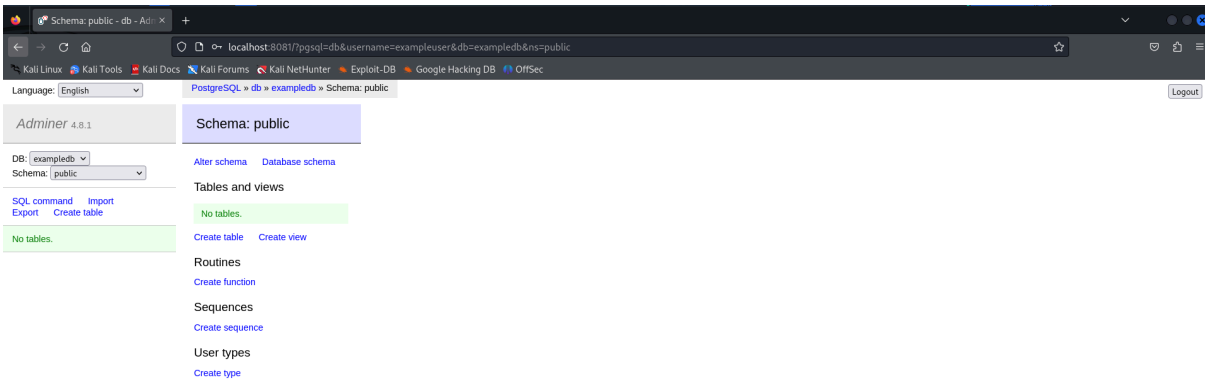
    - Ensure all containers are running by using: docker-compose ps
    - 

2. Testing the Web Server:
    - Open your browser and navigate to http://localhost:8080. You should see the Nginx welcome page.
    - 

3. Testing the Database and Adminer:
    - Open your browser and navigate to http://localhost:8081. You should see the Adminer login page.
    - Log in using the PostgreSQL credentials you defined in your Docker Compose file.



-
- Click on the MySQL dropdown to switch to PostgreSQL and them log into the database using the credentials mentioned in the docker-compose.yml file.

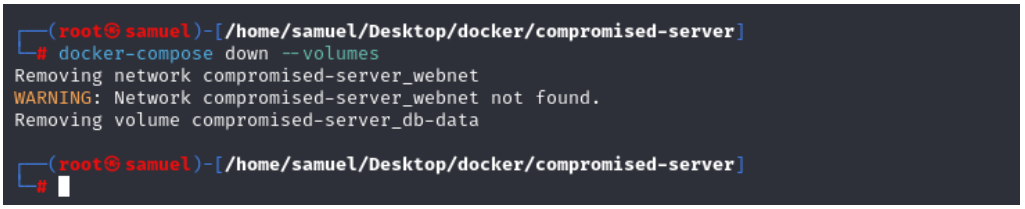


-

Task 3: Docker Compose Management

1. Stopping Services:
   - Stop your composed environment using:
     docker-compose down
   - Deliverable: Screenshot showing that the containers have been stopped and removed.

2. Cleanup:
   - Remove all stopped containers, networks, and volumes created by Docker Compose using:
     docker-compose down --volumes

- 
┌──(root㉿samuel)-[/home/samuel/Desktop/docker/compromised-server]
└─# docker-compose down --volumes
Removing network compromised-server_webnet
WARNING: Network compromised-server_webnet not found.
Removing volume compromised-server_db-data
┌──(root㉿samuel)-[/home/samuel/Desktop/docker/compromised-server]
└─#