# **eCR Now Setup Instructions using Docker Compose**

## **Prerequisites**

* Docker and Docker Compose installed on your machine.

## 

## **Setup with Docker Compose**

## **Steps**

#### **1.1 Create a Docker Compose File**

## **Create a file named docker-compose.yml in your project root directory. Here's an example configuration:**

services:

postgres:

image: postgres:15

container\_name: postgres\_container

environment:

POSTGRES\_PASSWORD: ecrnow@2024

POSTGRES\_DB: ecrnow

networks:

- ecrnow\_network

ports:

- "5432:5432"

ecr-now:

build:

context: .

dockerfile: Dockerfile

container\_name: ecr-now

depends\_on:

- postgres

networks:

- ecrnow\_network

volumes:

- D://ecr-now:/config

environment:

- logging.file.name=/config/ecrNow.log

- ersd.file.location=/config/eRSDv2\_specification\_bundle.json

- schematron.file.location=/config/schematron/CDAR2\_IG\_PHCASERPT\_R2\_STU1.1\_SCHEMATRON.sch

- xsd.schemas.location=/config/schemas/CDA\_SDTC.xsd

- kar.directory=/config/kars

- bsa.output.directory=/config/bsa-output/

- custom-query.directory=/config/custom-queries

- jdbc.url=jdbc:postgresql://postgres\_container:5432/ecrNow

- jdbc.username=postgres

- jdbc.password=ecr@2024

- security.key=test123

ports:

- "8081:8080"

networks:

ecrnow\_network:

driver: bridge

external: true

name: ecrnow\_network

**Note:** The volumes configuration in Docker Compose maps a directory on the host to a directory inside the container. In the example above, ./config refers to a relative path on the host system. You can adjust the path based on your operating system:

* Windows: Use a path like D://ecr-now/config or C:/Users/yourusername/ecr-now/config.
* Linux: Use a path like /home/yourusername/ecr-now/config or ./config for relative paths.

**Environment Variables:**

* Verify that the paths and values for the environment variables are correctly set according to your application’s configuration requirements.

#### 

#### 

#### 

#### **1.2 Create a Dockerfile**

Create a file named Dockerfile in the project root directory. The Dockerfile defines the image's contents and instructions for building it.

| FROM openjdk:17-alpine  RUN apk --no-cache add maven && mvn --version  WORKDIR /java-app  COPY pom.xml .  COPY src ./src  # Package the Spring Boot application, skipping tests  RUN mvn clean install -Dmaven.test.skip=true  # Define the command to run your application  CMD ["java", "-jar", "./target/ecr-now.war"] |
| --- |

#### **1.3 Build and Run the Containers**

Navigate to the directory containing the docker-compose.yml file and run the following command:

docker-compose up -d --build

This command will build the Docker image defined in the Dockerfile and start the containers as per the configuration in docker-compose.yml.

#### **1.4: Run PgAdmin Container**

*1)Run PgAdmin Container:*

docker run -d --network ecrnow\_network --name pgadmin-container -p 5050:80 -e PGADMIN\_DEFAULT\_EMAIL=user@domain.com -e PGADMIN\_DEFAULT\_PASSWORD=postgres -d dpage/pgadmin4

Remember to replace user@domain.com and postgres with your actual email and password.

This command runs PgAdmin in a Docker container (dpage/pgadmin4) on port 5050. It sets the default login credentials (user@domain.com and postgres).   
   
   
 2) Refer this link [Connecting PostgreSQL Container using PgAdmin 4](#kix.854ffr8ysuer)

#### **1.5 Run eCR Now UI**

Run the eCR Now UI container with the specified environment variable:

docker run -d --network ecrnow\_network --restart always -e REACT\_APP\_ECR\_BASE\_URL=http://localhost:8081 --name ecrNow-ui -p 3000:3000 drajerhealth/ecr-now:ecr-now-ui-3.1.4

**Go to browser and hit http://localhost:3000**

#### **1.6: Verify Setup**

After running the containers, you can verify the setup by accessing the following:

* **eCR Now Backend**: http://{{server\_ip\_address}}:8081
* **eCR Now UI**: http://{{server\_ip\_addres}}:3000

**2. Using Dockerfile Only**

### **Overview**

You can manually build and run Docker images using only a Dockerfile. This is useful for building and deploying single-container applications.

### **Prerequisites**

* Docker installed on your system.

### **Steps**

#### **2.1 Create a Dockerfile**

#### Create a file named Dockerfile in your project root directory. Here’s an example Dockerfile

| FROM openjdk:17-alpine  RUN apk --no-cache add maven && mvn --version  WORKDIR /java-app  COPY pom.xml .  COPY src ./src  # Package the Spring Boot application, skipping tests  RUN mvn clean install -Dmaven.test.skip=true  # Define the command to run your application  CMD ["java", "-jar", "./target/ecr-now.war"] |
| --- |

#### 

#### **2.2 Build the Docker Image**

### **Navigate to the directory containing the Dockerfile and run the following command:**

### **docker build -t ecr-now-image .**

#### **2.3 Create Docker Network**

Create a docker network on the name of ecrnow\_network**,** use below command:

docker network create ecrnow\_network

#### **2.4 : Run PostgreSQL Container**

Run the PostgreSQL container with the specified network and environment variables, use below command:

docker run -d --network ecrnow\_network --name postgres\_container --restart always -e POSTGRES\_PASSWORD=ecrnow@2024 -p 5432:5432 postgres:15

Note: Add password of your choice

#### **2.5: Run PgAdmin Container**

*1)Run PgAdmin Container:*

docker run -d --network ecrnow\_network --name pgadmin-container -p 5050:80 -e PGADMIN\_DEFAULT\_EMAIL=user@domain.com -e PGADMIN\_DEFAULT\_PASSWORD=postgres -d dpage/pgadmin4

Remember to replace user@domain.com and postgres with your actual email and password.

This command runs PgAdmin in a Docker container (dpage/pgadmin4) on port 5050. It sets the default login credentials (user@domain.com and postgres).   
   
   
 2) Refer this link [Connecting PostgreSQL Container using PgAdmin 4](#kix.854ffr8ysuer)

#### **2.6: Create ecrnow Database**

### **If the “ ecrnow” database is created using PgAdmin in Step 2.5 , you can skip this step.**

### **Execute the following command to create the ecrnow database inside the PostgreSQL container**

### **docker exec -it postgres\_container psql -U postgres -c "CREATE DATABASE ecrnow;"**

#### 

#### **2.7 Run eCR Now Backend**

### 

docker run -d --network ecrnow\_network -v D://ecr-now:/config -e logging.file.name=/config/ecrNow.log -e ersd.file.location=/config/eRSDv2\_specification\_bundle.json -e schematron.file.location=/config/schematron/CDAR2\_IG\_PHCASERPT\_R2\_STU1.1\_SCHEMATRON.sch -e xsd.schemas.location=/config/schemas/CDA\_SDTC.xsd -e kar.directory=/config/kars -e bsa.output.directory=/config/bsa-output/ -e custom-query.directory=/config/custom-queries -e jdbc.url=jdbc:postgresql://postgres\_container:5432/ecrnow -e jdbc.username=postgres -e jdbc.password=ecrnow@2024 -e security.key=test123 --link postgres\_container:postgres -p 8081:8081 --name ecr-now ecr-now-image

**Note:** The volumes configuration in Docker Compose maps a directory on the host to a directory inside the container. In the example above, ./config refers to a relative path on the host system. You can adjust the path based on your operating system:

* Windows: Use a path like D://ecr-now/config or C:/Users/yourusername/ecr-now/config.
* Linux: Use a path like /home/yourusername/ecr-now/config or ./config for relative paths.

**Environment Variables:**

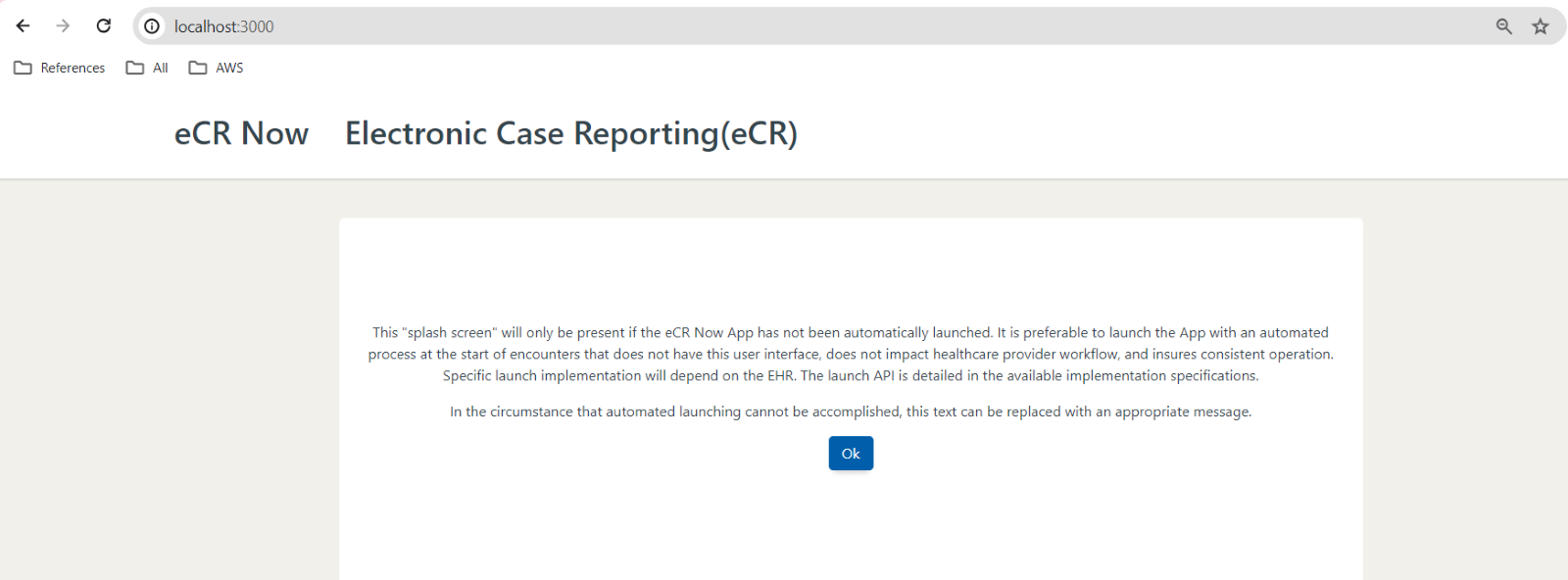
* Verify that the paths and values for the environment variables are correctly set according to your application’s configuration requirements.

#### **2.8 Run eCR Now UI**

Run the eCR Now UI container with the specified environment variable:

docker run -d --network ecrnow\_network --restart always -e REACT\_APP\_ECR\_BASE\_URL=http://localhost:8081 --name ecrNow-ui -p 3000:3000 drajerhealth/ecr-now:ecr-now-ui-3.1.4

**Go to browser and hit http://localhost:3000**



#### **2.9: Verify Setup**

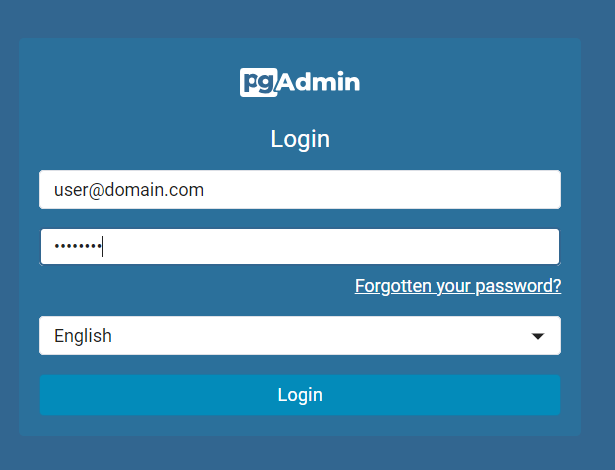
After running the containers, you can verify the setup by accessing the following:

* **eCR Now Backend**: http://{{server\_ip\_address}}:8081
* **eCR Now UI**: http://{{server\_ip\_addres}}:3000

Connecting to Database Container using pgAdmin 4

Log in to pgAdmin 4

Once the container is successfully running (if you encounter any issues, it’s a good idea to check the Docker Desktop app to ensure the container is running), you can access pgAdmin by navigating to localhost:5050 in a web browser of your choice.



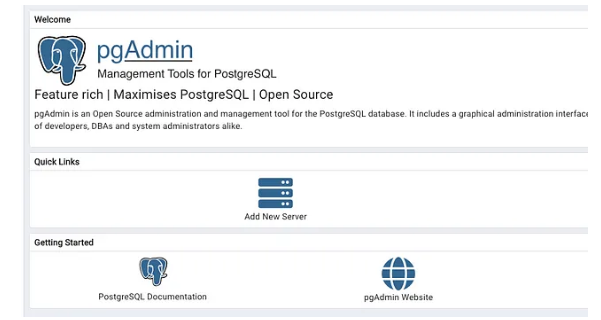
You will then see a login prompt; you will be able to log in with the e-mail address and password that you specified previously when running the pg Admin Docker container

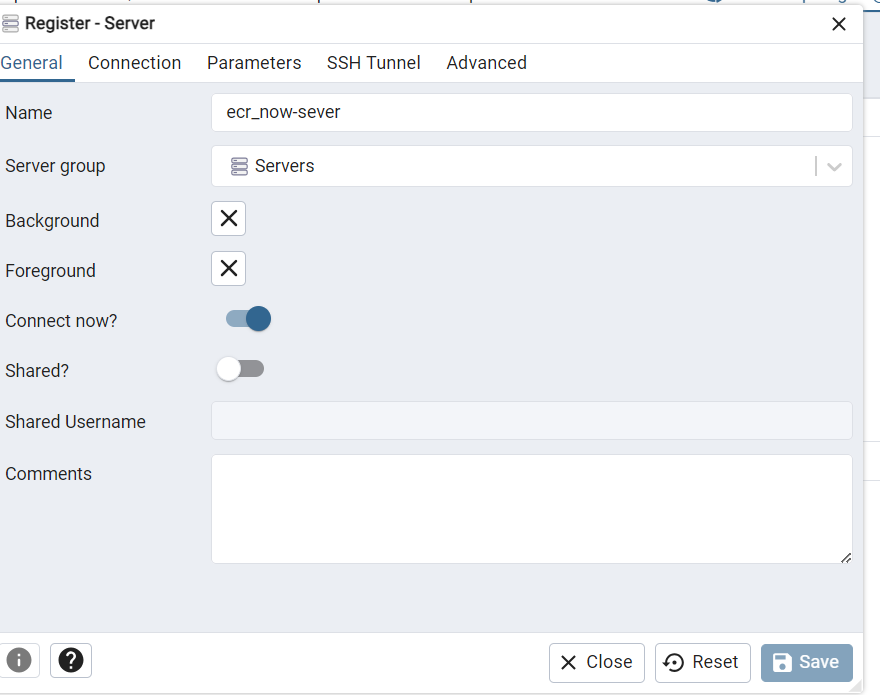
, in our case “user@domain.com” and “postgres”.

Note :-The username and password should be the same as those used when running the pg Admin Docker container.

Connect to Database Container/ Add Server

In the next step, we are going to connect to the database container. For this, you need to click on Add New Server:



And enter the relevant information to connect to our database, in the Name field we can choose an Alias to refer to our database in Pg Admin:   


Before we proceed, it’s important to obtain the IP address of the “postgres” container. To find the IP address, you can execute the following command in your terminal (Linux/macOS) or PowerShell (Windows):

docker inspect -f '{{.NetworkSettings.Networks.ecrnow\_network.IPAddress}}' postgres\_container

In our case, it returned the IP address 172.17.0.3 (this may or may not be true for you). Now we enter all the connection information:

Host name/address: 172.17.0.3 (might be different for you)   
 or   
Host name/address: postgres\_container (name of Postgres container)

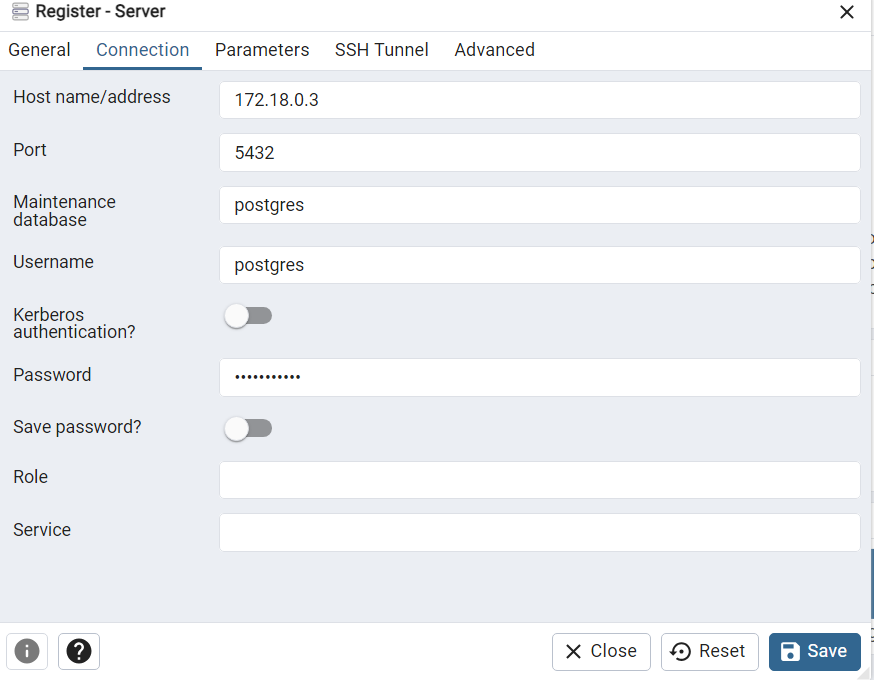
Port: 5432 (port of postgres container)

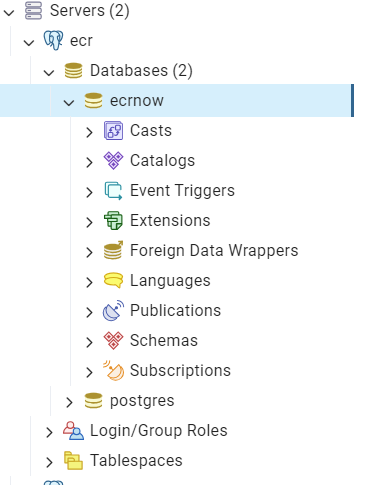
Maintenance database: postgres

Username: postgres (or another name if you changed it)

Password: ecrnow@2024 (or whatever password you selected for Postgres container)

Optional: Set save password to true



Click Save and you will be able to select your database server from the menu on the left side:   
   
   
   


Note:-Create the ecrnow database if database does not exist

Congratulations, now your environment should be up and running. For questions about pgAdmin 4 refer to the documentation:   
[PgAdmin Docs](https://www.pgadmin.org/docs/pgadmin4/development/index.html?source=post_page-----4a8d81048aea--------------------------------)