Andy N., Anthony P., Dawn D., Divyanshu F., Huan L.

InSTALLATION & DEPLOYMENT GUIDE

Software setup and application deployment for Decision Support System

Table of Contents

[1 Introduction 2](#_Toc184386669)

[2 Prerequisites 3](#_Toc184386670)

[3 Docker Installation 3](#_Toc184386671)

[4 Planning Management System (PMS) 4](#_Toc184386672)

[4.1 PMS Database 4](#_Toc184386673)

[4.2 PMS Application 5](#_Toc184386674)

[5 Data Model 7](#_Toc184386675)

[6 Airflow for Data Pipelines (ETL) 9](#_Toc184386676)

[7 Metabase for Dashboards 10](#_Toc184386677)

[7.1 Metabase Installation on Docker 10](#_Toc184386678)

[7.2 Import Metabase Report 11](#_Toc184386679)

# Introduction

This document provides a comprehensive guide to set up and deploy a **Decision Support System** (DSS) for **Springboard Clinics**, leveraging modern data tools and technologies. The primary goal of this system is to enhance clinic operations through streamlined data management, efficient ETL (Extract, Transform, Load) processes, and intuitive reporting dashboards.

A diagram of a data processing process

Description automatically generated

The system architecture, as illustrated, is designed to integrate seamlessly with existing tools, ensuring scalability, reliability, and ease of use. Key components of this architecture include:

1. **Planning Management System**:
   * Developed using **Streamlit** for user-friendly planning and management.
   * A robust **PostgreSQL** database serves as the foundation for storing and managing clinic staff planning and operational data.
   * Deployed via **Docker** for simplified setup and maintenance.
2. **ETL Pipeline**:
   * **Apache Airflow** automates the extraction and transformation of data from the clinic's **Jane System** and planning database.
   * The pipeline ensures data consistency and accuracy, loading it into a data model tailored for performance monitoring and reporting.
3. **Performance Dashboards**:
   * A comprehensive dashboard built with **Metabase** provides insights into clinic performance, staff efficiency, and target achievement.
   * The data model, hosted on **PostgreSQL**, supports these reports with a well-structured schema optimized for analytical queries.

The deployment of this architecture relies on **Docker**, ensuring that each component operates in isolated and consistent environments, facilitating a smooth setup, updates, and scalability. By implementing this decision support system, Springboard Clinics will gain actionable insights into their operations, empowering data-driven decisions and enhancing overall performance.

# Prerequisites

* **Operating System**: Windows 10 or higher.
* **RAM**: Minimum 8 GB (16 GB recommended).
* **CPU**: Minimum 4 cores.
* **Software**:
  + **Docker Desktop**: Latest version.
  + **Docker Images:**
    - postgres:latest
    - python:3.9-slim
    - extending\_airflow:latest
    - metabase:latest

# Docker Installation

**Download Docker Desktop**:

* Visit [Docker's official website](https://docs.docker.com/compose/install/) and download the installer for Windows.

**Install Docker Desktop**:

* Run the downloaded installer.
* Follow the prompts to complete the installation.

**Verify Installation**:

* Open a terminal and run:

docker --version

Docker version 24.0.6, build ed223bc

* Ensure Docker is running before proceeding.

# Planning Management System (PMS)

## PMS Database

**PostgreSQL Deployment on Docker**

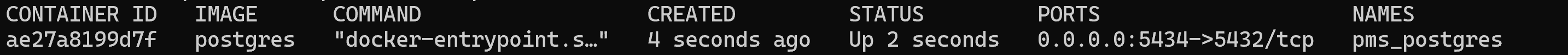
1. Run the command to configure PostgreSQL for the **Planning Management System**:

docker pull postgres

docker run --name pms\_postgres -e POSTGRES\_PASSWORD=your\_password -p 5434:5432 -d postgres

Verify that the database container is running:

docker ps -f "name=pms\_\*"



1. Create table structure on pms database:

Connect to database:

docker exec -it pms\_postgres psql -U postgres

Execute scripts to create tables:

-- Create the database

CREATE DATABASE springboard;

-- Connect to the database

\c springboard;

-- Create the schema

CREATE SCHEMA IF NOT EXISTS planning1;

-- Create the practitioner table

CREATE TABLE IF NOT EXISTS planning1.practitioner (

practitioner\_id INTEGER NOT NULL,

practitioner\_name CHARACTER VARYING(255),

employee\_type CHARACTER VARYING(255),

clinic\_location CHARACTER VARYING,

bill\_rate\_standard INTEGER,

bill\_rate\_special INTEGER,

manager\_name CHARACTER VARYING(50),

CONSTRAINT practitioner\_pkey PRIMARY KEY (practitioner\_id)

);

-- Create the statutory\_holidays table

CREATE TABLE IF NOT EXISTS planning1.statutory\_holidays (

holiday\_date DATE NOT NULL,

holiday\_name CHARACTER VARYING(100),

CONSTRAINT statutory\_holidays\_pkey PRIMARY KEY (holiday\_date)

);

-- Create the target\_update table

CREATE TABLE IF NOT EXISTS planning1.target\_update (

practitioner\_id INTEGER,

practitioner\_name CHARACTER VARYING(255),

target\_date TIMESTAMP,

target\_hour DOUBLE PRECISION,

updated\_at TIMESTAMP,

CONSTRAINT unique\_practitioner\_date UNIQUE (practitioner\_id, target\_date)

);

## PMS Application

List of files for application deployment:

A screenshot of a computer

Description automatically generated

* .**env**: store database connection credentials
* **Dockerfile**: to build docker image
* **Requirements.txt**: to list out python libraries
* **Target\_update\_final.py**: application code

This is the content of Dockerfile and requirement.txt

Dockerfile

# Use the official Python image

FROM python:3.9-slim

# Set the working directory

WORKDIR /app

# Copy the application code

COPY . /app

# Install system dependencies

RUN apt-get update && apt-get install -y libpq-dev gcc

# Install Python dependencies

RUN pip install --no-cache-dir -r requirements.txt

# Expose Streamlit default port

EXPOSE 8501

# Set Streamlit environment variables

ENV STREAMLIT\_SERVER\_PORT=8501

ENV STREAMLIT\_SERVER\_ADDRESS=0.0.0.0

# Command to run the app

CMD ["streamlit", "run", "target\_update\_final.py"]

requirement.txt

streamlit

pandas

psycopg2-binary

matplotlib

streamlit-option-menu

python-dotenv

Build docker image: run cmd at source code folder

docker build -t pms-app .

A computer screen shot of a computer program

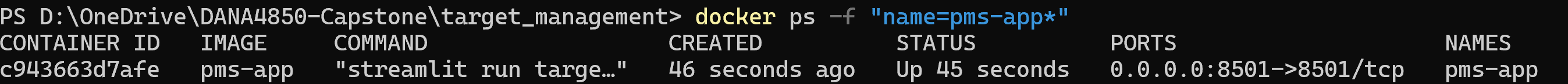
Description automatically generated

Run container to deploy app:

docker run -d -p 8501:8501 --name pms-app pms-app

Verify if the container is running:

docker ps -f "name=pms-app\*"



Test planning management app on web browser:

A screenshot of a computer

Description automatically generated

# Data Model

To save resources, the data model is also deployed on pms database with a different schema.

Connect to database:

docker exec -it pms\_postgres psql -U postgres

Execute scripts to create tables:

-- Connect to the database

\c springboard;

-- Create the schema

CREATE SCHEMA IF NOT EXISTS model;

-- Create the Dim\_Date table in PostgreSQL

CREATE TABLE model.Dim\_Date (

date DATE PRIMARY KEY,

day\_of\_week VARCHAR(10),

week\_of\_year INT,

month INT,

quarter INT,

year INT

);

-- Populate the Dim\_Date table with dates from 2020-01-01 to 2030-12-31

INSERT INTO model.Dim\_Date (date, day\_of\_week, week\_of\_year, month, quarter, year)

SELECT

d::DATE AS date,

TO\_CHAR(d, 'Day') AS day\_of\_week,

EXTRACT(WEEK FROM d)::INT AS week\_of\_year,

EXTRACT(MONTH FROM d)::INT AS month,

EXTRACT(QUARTER FROM d)::INT AS quarter,

EXTRACT(YEAR FROM d)::INT AS year

FROM generate\_series('2020-01-01'::DATE, '2030-12-31'::DATE, INTERVAL '1 day') AS d;

CREATE TABLE model.Dim\_Practitioner (

practitioner\_dim\_id SERIAL PRIMARY KEY,

practitioner\_id INT NOT NULL,

practitioner\_name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

contract\_type VARCHAR(50),

manager\_name VARCHAR(100),

location\_id INT,

location\_name VARCHAR(100),

effective\_start\_date DATE,

effective\_end\_date DATE,

is\_current BOOLEAN DEFAULT TRUE

);

CREATE TABLE model.Dim\_Item (

item\_id SERIAL PRIMARY KEY,

item\_name VARCHAR(100) NOT NULL,

department VARCHAR(100),

category VARCHAR(100)

);

-- Create the Dim\_Location table in PostgreSQL

CREATE TABLE model.Dim\_Location (

location\_id SERIAL PRIMARY KEY,

location\_name VARCHAR(100) NOT NULL,

address VARCHAR(255)

);

CREATE TABLE model.Fact\_Performance (

date DATE NOT NULL,

practitioner\_dim\_id INT NOT NULL,

location\_id INT NOT NULL,

target\_hour DECIMAL(5, 1),

actual\_hour DECIMAL(5, 1),

total\_billing DECIMAL(10, 1)

);

CREATE TABLE model.Fact\_Appointments (

date DATE NOT NULL,

practitioner\_dim\_id INT NOT NULL,

location\_id INT NOT NULL,

item\_id INT,

number\_appoiments INT,

actual\_hour DECIMAL(5, 1),

total\_billing DECIMAL(10, 1)

);

# Airflow for Data Pipelines (ETL)

We have the airflow source code project as follows:

A screenshot of a computer

Description automatically generated

Main components are:

* **Dags** folder: store all the data pipelines source code
* **Docker-compose.yaml**: declare all the services to run airflow
* **Requirements.txt**: declare all python libraries used

Build airflow on docker: run cmd in the source code project

docker-compose up airflow-init

docker-compose up -d

Verify if all the airflow containers are running:

A black screen with white text

Description automatically generated

Verify on web browser: go to <http://localhost:8080>

A screenshot of a computer

Description automatically generated

Setup database connection for Airflow:

Go to **Admin** à **Connections**:

A screenshot of a computer

Description automatically generated

Add new connections or edit existing connections and save.

# Metabase for Dashboards

## Metabase Installation on Docker

1. Use docker-compose.yaml to configure Metabase:

version: "3.8"

services:

metabase:

image: metabase/metabase:latest

container\_name: metabase

ports:

- "3000:3000"

volumes:

- metabase\_data:/metabase-data

volumes:

metabase\_data:

1. Start the Metabase service:

docker-compose up -d metabase

Verify if the docker container is running:



Verify on web browser: <http://localhost:3000/>

A screenshot of a computer

Description automatically generated

## Import Metabase Report

1. Replace the metabase.mv.db file in the folder /metabase.db in the container with the file provided from the development environment.
2. Restart Metabase:

docker restart metabase

1. Access Metabase at http://localhost:3000 and verify the imported reports.

A screenshot of a computer

Description automatically generated