

# Week 7: Docker + Docker Compose

## CSI403 Full Stack Development | Lab 6 (8%)

Semester 1/2569

# Agenda

- 1 Docker Introduction
- 2 Dockerfile
- 3 Docker Compose
- 4 Lab 6 Assignment

# The Problem

“It works on my machine!”

## Common issues:

- Different OS versions
- Different Python versions
- Missing dependencies
- Environment variable differences
- Database configuration mismatches

## Package your app with everything it needs

- Same environment everywhere
- Isolated from host system
- Easy to deploy
- Easy to scale
- Easy to share

# Docker Concepts



# Image vs Container

## Image:

- Blueprint/template
- Read-only
- Contains app + dependencies
- Can create many containers
- Stored in registry

## Container:

- Running instance
- Read-write layer
- Has its own filesystem
- Has its own network
- Can be started/stopped

# What is Dockerfile?

**Text file with instructions to build an image**

- FROM - Base image
- WORKDIR - Working directory
- COPY - Copy files
- RUN - Execute commands
- EXPOSE - Expose ports
- CMD - Default command

# Dockerfile for TaskFlow

```
# Base image
FROM python:3.11-slim

# Set working directory
WORKDIR /app

# Install system dependencies
RUN apt-get update && apt-get install -y \
    curl gnupg2 unixodbc-dev \
&& rm -rf /var/lib/apt/lists/*

# Copy and install Python dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy application code
COPY .

# Expose port
EXPOSE 8000

# Run command
```

# Build Image

```
# Build image with tag
docker build -t taskflow:latest .

# Build with specific Dockerfile
docker build -f Dockerfile.prod -t taskflow:prod .

# List images
docker images
```

# Run Container

```
# Run container
docker run -d -p 8000:8000 --name taskflow-app taskflow:latest

# Options explained:
# -d           Run in background (detached)
# -p 8000:8000 Map host port to container port
# --name       Give container a name

# View running containers
docker ps

# View all containers (including stopped)
docker ps -a
```

# Container Management

```
# View logs
docker logs taskflow-app
docker logs -f taskflow-app # Follow logs

# Execute command in container
docker exec -it taskflow-app bash

# Stop container
docker stop taskflow-app

# Start stopped container
docker start taskflow-app

# Remove container
docker rm taskflow-app
```

## .dockerignore

```
# .dockerignore
venv/
--pycache__/
*.pyc
.git/
.gitignore
.env
*.md
tests/
.pytest_cache/
```

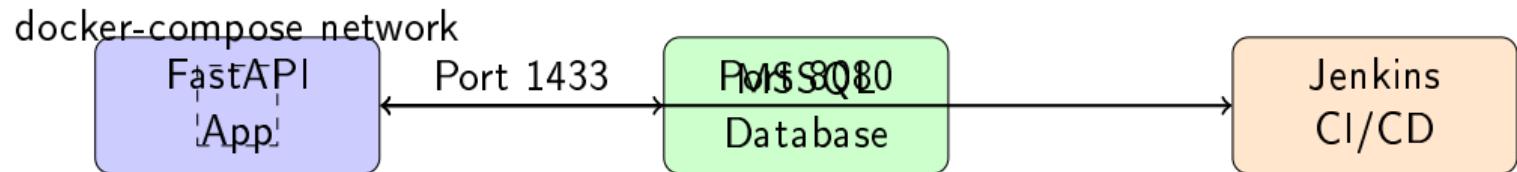
Prevents unnecessary files from being copied to image

# What is Docker Compose?

**Tool for defining multi-container applications**

- Define all services in one file
- Start everything with one command
- Automatic networking between containers
- Easy to manage dependencies

# Our Architecture



# docker-compose.yml: Version and App

```
version: '3.8'

services:
  app:
    build: .
    container_name: taskflow-app
    ports:
      - "8000:8000"
    environment:
      - DATABASE_URL=${DATABASE_URL}
    depends_on:
      - db
  networks:
    - taskflow-network
  restart: unless-stopped
```

# docker-compose.yml: Database

```
db:
  image: mcr.microsoft.com/mssql/server:2022-latest
  container_name: taskflow-db
  environment:
    - ACCEPT_EULA=Y
    - SA_PASSWORD=${DB_PASSWORD}
  ports:
    - "1433:1433"
  volumes:
    - mssql_data:/var/opt/mssql
  networks:
    - taskflow-network
  restart: unless-stopped
```

## docker-compose.yml: Jenkins

```
jenkins:
  image: jenkins/jenkins:lts
  container_name: taskflow-jenkins
  ports:
    - "8080:8080"
    - "50000:50000"
  volumes:
    - jenkins_home:/var/jenkins_home
  networks:
    - taskflow-network
  restart: unless-stopped
```

# docker-compose.yml: Networks and Volumes

```
networks:  
  taskflow-network:  
    driver: bridge  
  
volumes:  
  mssql_data:  
  jenkins_home:
```

**Networks:** Allow containers to communicate

**Volumes:** Persist data even when containers are removed

# Environment Variables (.env)

```
# .env file
DB_PASSWORD=YourStrong@Pass123
DATABASE_URL=mssql+pyodbc://sa:YourStrong@Pass123@db:1433/taskflow?driver=ODBC+
    Driver+17+for+SQL+Server
SECRET_KEY=your-secret-key-here
DEBUG=true
```

**Important:** Never commit .env to Git!

# Docker Compose Commands

```
# Start all services
docker-compose up -d

# View running services
docker-compose ps

# View logs
docker-compose logs -f
docker-compose logs app

# Stop all services
docker-compose down

# Rebuild and restart
docker-compose up -d --build
```

# More Compose Commands

```
# Stop without removing
docker-compose stop

# Start stopped services
docker-compose start

# Restart services
docker-compose restart

# Remove everything (including volumes)
docker-compose down -v

# Execute command in service
docker-compose exec app bash
```

# Lab 6: Docker + Compose (8%)

## Requirements:

- ① Create Dockerfile for TaskFlow
- ② Create .dockerignore
- ③ Create docker-compose.yml with 3 services
- ④ Create .env file for configuration
- ⑤ Verify everything works with docker-compose up
- ⑥ Document setup in README.md

# Lab 6 Grading Rubric

Criteria	Points
Dockerfile + .dockerignore	2%
docker-compose.yml (3 services)	3%
Networks + Volumes	1.5%
App connects to database	1%
Documentation	0.5%
<b>Total</b>	<b>8%</b>

# Questions?

Containerize your application!

Next Week: Testing + Jenkins CI