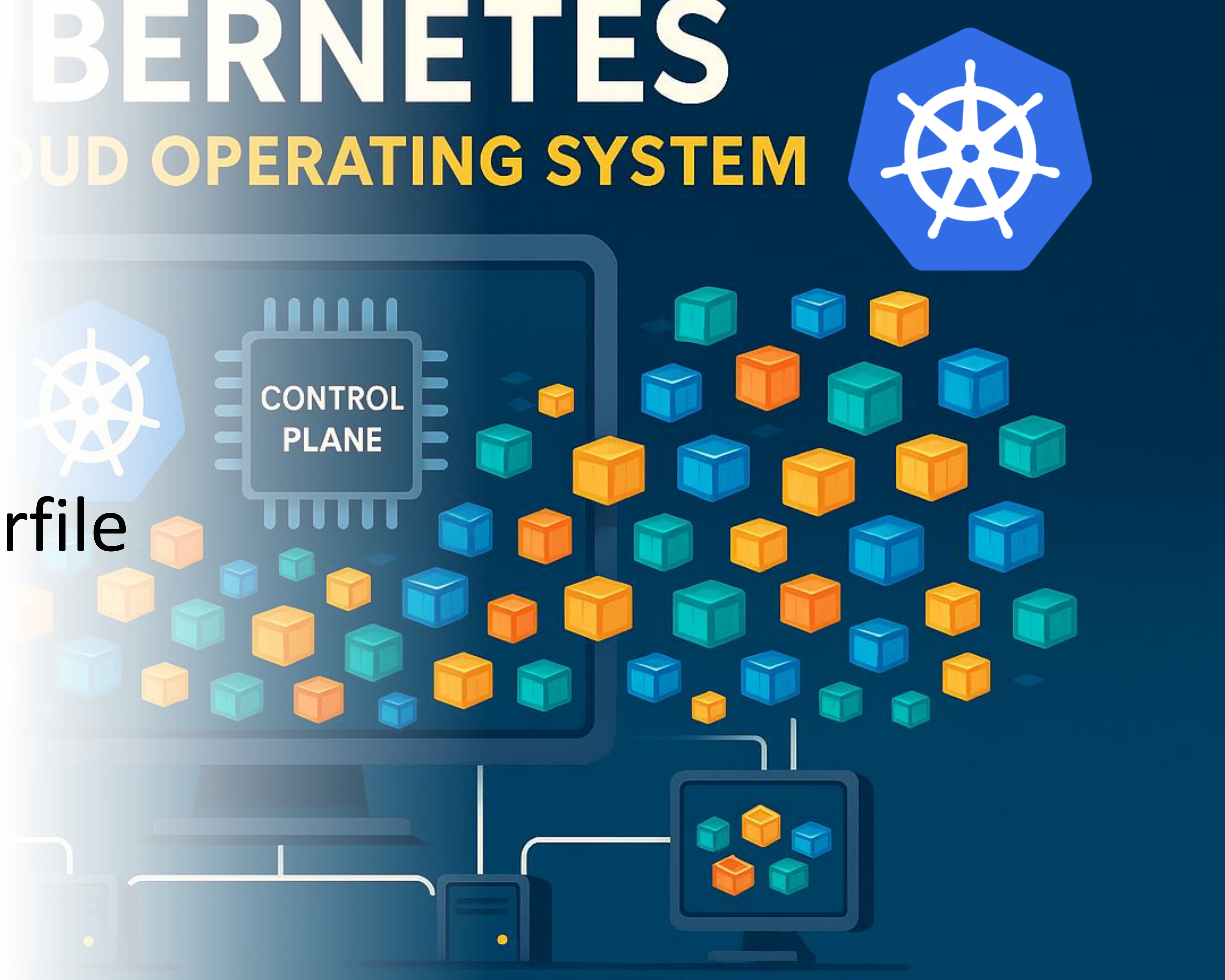
CS-UY 3913
Container
Operating Systems



# Building Dockerfile

Instructor: Magdy Salem

## Agenda

- Dockerfile Overview
- Advance Syntax
- Multi Stage Build
- Demo
- Lab

# Dockerfile Overview

- **Dockerfile** is a simple, declarative text file that acts like a recipe for building a Docker image.
- It defines all the steps like setting a base image, installing dependencies, copying code, exposing ports, and specifying the startup process
- Making the build repeatable, version-controlled, and consistent across environments.

## Key concepts

- **Base Image:** You start with a minimal OS or language runtime (e.g., `FROM python:3.9-slim`) so you don't reinvent the wheel.
- **Build Instructions:** Lines like `RUN apt-get update && apt-get install -y git` or `COPY ./app /usr/src/app` layer in packages, code, and configuration.
- **Metadata & Defaults:** ENV and ARG set build-time or runtime variables; EXPOSE documents which ports your app will listen on; ENTRYPOINT/CMD define the default process.
- **Layered Image Model:** Each instruction creates a new layer; smart caching means unchanged steps are skipped on rebuild, speeding development iterations.

## How it fits into  workflow

- **Local development:** Build (`docker build`) and run (`docker run`) images to test changes quickly.
- **CI/CD pipelines:** Automate image builds, run tests inside containers, and push to a registry.
- **Production deployments:** Orchestrators like Kubernetes pull your immutable images, guaranteeing consistency across staging and prod environments.

## Dockerfile Syntax

- FROM
- RUN
- CMD
- ENTRYPOINT
- COPY
- ADD
- WORKDIR

## Dockerfile Syntax

**ENV**

- **Purpose:** Sets environment variables inside the container.
- **Example:**

```
   dockerfile
CopyEdit
ENV NODE_ENV=production
```

# Dockerfile Syntax

**ARG**

- **Purpose:** Defines build-time variables (not present in the final container).
- **Example:**

  dockerfile

CopyEdit

```
ARG VERSION=1.0
RUN echo "Building version $VERSION"
```

## Dockerfile Syntax

**EXPOSE**

- **Purpose:** Documents the port the container listens on.
- **Example:**

```
   dockerfile
CopyEdit
EXPOSE 8080
```

## Dockerfile Syntax

**VOLUME**

- **Purpose:** Declares a mount point for external volumes.
- **Example:**

```
    dockerfile
CopyEdit
VOLUME /data
```

## Dockerfile Syntax

**USER**

- **Purpose:** Specifies the user under which the container should run.
- **Example:**

```
    dockerfile
CopyEdit
USER appuser
```

## Dockerfile Syntax

**HEALTHCHECK**

- **Purpose:** Defines how Docker should test container health.
- **Example:**

  dockerfile

CopyEdit

```
HEALTHCHECK CMD curl --fail
  http://localhost:8080/health || exit 1
```

## Dockerfile Best Practice

- **Minimize Layers**
- **Use `.dockerignore`**
- **Optimize Caching**

## Multi-Stage Builds

- Allows separating build-time and runtime environments.
- Reduces final image size by excluding dev tools and dependencies.

```
# Stage 1: Build
FROM golang:1.20 AS builder
WORKDIR /app
COPY . .
RUN go build -o app

# Stage 2: Runtime
FROM alpine
COPY --from=builder /app/app /usr/bin/app
ENTRYPOINT ["app"]
```

## Dockerfile Syntax

Read more about the concept at Docker Docs

https://docs.docker.com/build/concepts/dockerfile/

## Demo

# LAB

Lab Github link here