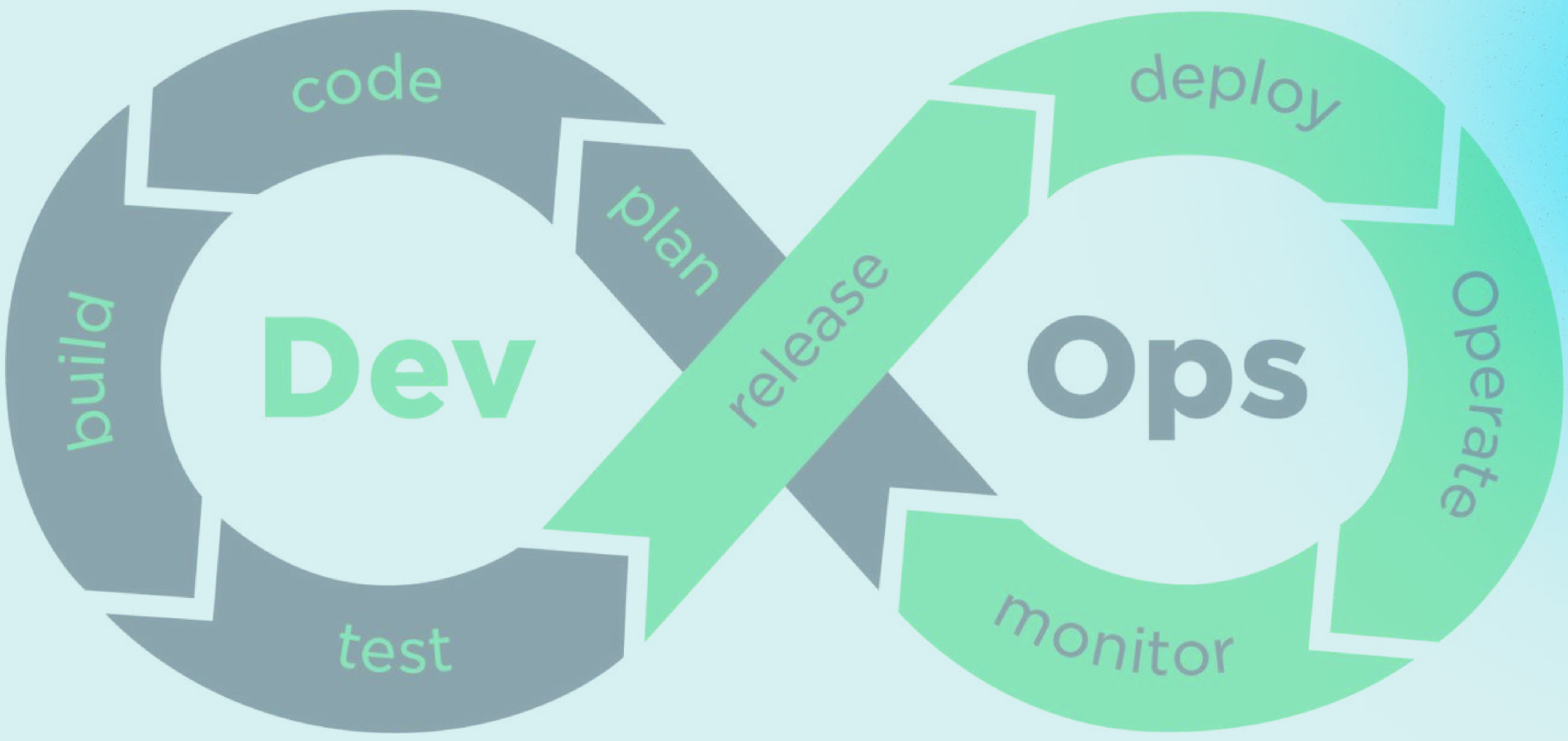
Dockerfiles for Well known Stacks



Dockerfile for Node.js

```dockerfile
# Use the official Node.js image from the Docker Hub
FROM  node:22-alpine

# Set the working directory
WORKDIR  /app

# Copy the package.json and package-lock.json files
COPY  package*.json   ./

# Install the dependencies
RUN  npm install

# Copy the rest of the application code
COPY   . .

# Expose the application port
EXPOSE 3000

# Run the application
CMD ["npm", "start"]
```

Dockerfile for Java

```
# Use the official OpenJDK image from the Docker Hub
FROM  openjdk:18-jdk-alpine3.15

# Set the working directory
WORKDIR  /app

# Copy the jar file into the container (eg.  myapp.jar)
COPY target/myapp.jar  ./

# Run the application
CMD  ["java", "-jar", "myapp.jar"]

# Expose the application port
EXPOSE  8080
```

Dockerfile for Python

```
# Use the official Python image from the Docker Hub
FROM  python:alpine3.19

# Set the working directory
WORKDIR  /app

# Copy the requirements file into the container
COPY requirements.txt .

# Install needed packages specified in requirements.txt
RUN pip install -r requirements.txt

# Copy the rest of the application code
COPY  .  .
# Run the application
CMD ["python", "app.py"]
```

Dockerfile for Golang

```dockerfile
# Use the official Golang image from the Docker Hub
FROM   golang:alpine3.20

# Set the working directory
WORKDIR   /app

# Copy the source code
COPY   .  .

# Build the application
RUN  go build -o myapp  .

# Run the application
CMD   ["./myapp"]
```

Dockerfile for .NET

```
# Use the official .NET SDK image from the Docker Hub
FROM mcr.microsoft.com/dotnet/sdk:5.0
AS build

# Set the working directory
WORKDIR   /app

# Copy the csproj file and restore the dependencies
COPY *.csproj  ./

# downloads the NuGet packages specified in the .csproj
RUN  dotnet restore

# Copy the rest of the application code
COPY   .   .

# Build the application
RUN  dotnet publish -c Release -o /app


# Use the official .NET runtime image
FROM mcr.microsoft.com/dotnet/aspnet:5.0

# Set the working directory
WORKDIR   /app

# Copy the built files
COPY  --from=build /app .

# Run the application
ENTRYPOINT  ["dotnet", "myapp.dll"]
```

Dockerfile for PHP

```
# Use the official PHP image from the Docker Hub
FROM  php:8.2-cli

# Set the working directory
WORKDIR  /usr/src/myapp

# Copy the source code
COPY  .  /usr/src/myapp

# Run the application
CMD [ "php", "./your-script.php" ]

# Expose the application port
EXPOSE 80
```

Dockerfile for RUST

```
# Use a minimal Rust image
FROM rust:1.67-alpine

# Set the working directory
WORKDIR  /app

# Copy the Cargo.toml and Cargo.lock files
COPY  Cargo.*   ./

# Install dependencies
RUN cargo build --release

# Copy the rest of the application code
COPY   . .

# Expose the application port
EXPOSE 8080

# Command to run the application
CMD ["./target/release/your_app"]
```

Multistage Dockerfile for RUST

```
# Build stage--------------------

    # Use a Rust image as the build stage for efficiency
    FROM rust:1.67-alpine AS builder

    # Set the working directory
    WORKDIR   /app

    # Copy dependency files for management
    COPY Cargo.*   ./

    # Build the application with optimizations for performance
    RUN  cargo build --release

# Production stage

  # Use a minimal base image for production environment
  FROM alpine:latest

  # Set the working directory
  WORKDIR   /app

  # Copy the compiled binary from the build stage
  COPY --from=builder /app/target/release/* .

  # Expose port 8080 for external access (adjust as needed)
  EXPOSE 8080

  # Command to run the application
  CMD ["./target/release/your_app"]
```

Dockerfile for C++

```
# Use a Debian-based image with build tools
FROM  debian:buster-slim

# Install required packages
RUN  apt-get update && \
apt-get install -y build-essential g++

# Set the working directory
WORKDIR  /app

# Copy source code
COPY  . .

# Build the application
RUN   g++ main.cpp -o app

# Expose the port (if applicable)
EXPOSE  8080

# Command to run the application
CMD  ["./app"]
```

Dockerfile for RUBY

```
# Use the official Ruby image from the Docker Hub
FROM ruby:2.7-slim

# Set the working directory
WORKDIR /app

# Copy the Gemfile and Gemfile.lock
COPY Gemfile* ./

# Install the dependencies
RUN bundle install

# Copy the rest of the application code
COPY . .

# Expose the application port
EXPOSE  3000

# Command to run the application
CMD ["ruby", "app.rb"]
```