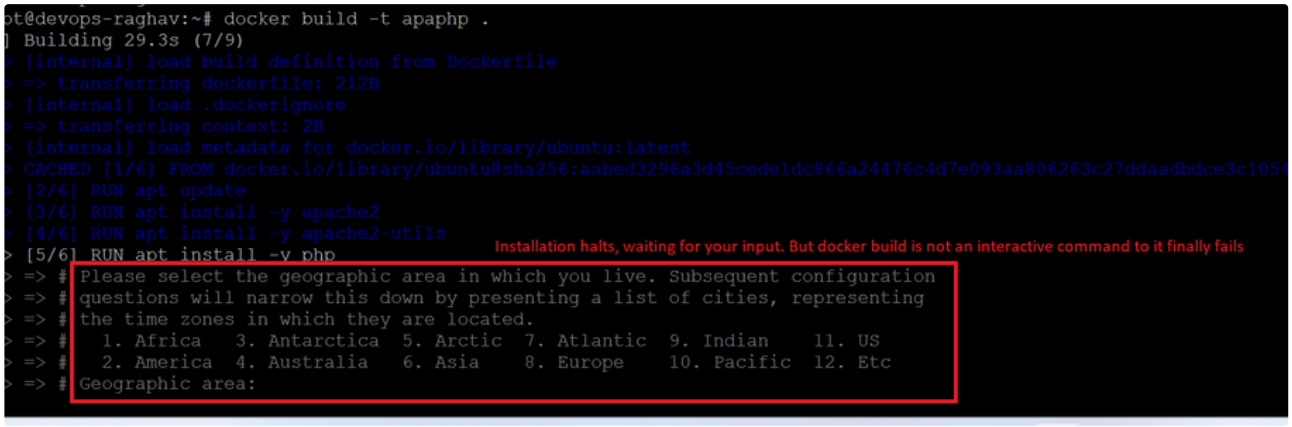
Dockerfile examples

Dockerfile for simple Apache Server

```
FROM ubuntu
RUN apt update
RUN apt install -y apache2
RUN apt install -y apache2-utils
RUN apt clean
EXPOSE 80
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Dockerfile to install Apache and PHP

```
FROM ubuntu
RUN apt update
RUN apt install -y apache2
RUN apt install -y apache2-utils
RUN apt install -y php
RUN apt clean
EXPOSE 80
CMD ["apache2ctl", "-D", "FOREGROUND"]
```



Modified Dockerfile to install Apache and PHP

```
FROM ubuntu
ENV TZ=Asia/Kolkata
RUN apt-get update
RUN apt-get install -y tzdata
RUN apt-get install -y apache2
RUN apt-get install -y apache2-utils
RUN apt-get install git -y
RUN apt update
RUN apt install -y libz-dev libssl-dev libcurl4-gnutls-dev libexpat1-dev gettext cmake gcc
RUN apt-get install -y nano && \
    apt-get install -y wget && \
    rm -fr /var/lib/apt/lists/*
RUN apt-get clean
```

```
EXPOSE 80
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Dockerfile for an Ubuntu-based LAMP (Linux, Apache, MySQL, PHP) stack

```
# Use the official Ubuntu image as the base
FROM ubuntu:20.04

# Set environment variables to avoid interactive prompts during installation
ENV DEBIAN_FRONTEND=noninteractive

# Install required packages for LAMP stack
RUN apt-get update && apt-get install -y \
    apache2 \
    mysql-server \
    php \
    php-mysql \
    php-apcu \
    libapache2-mod-php \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# Configure Apache web server
RUN a2enmod rewrite
RUN echo 'ServerName localhost' >> /etc/apache2/apache2.conf

# Start services
CMD ["apache2ctl", "-D", "FOREGROUND"]

# Expose ports
EXPOSE 80 3306
```

In this Dockerfile:

1. We start with the official Ubuntu 20.04 image as the base.
2. We set an environment variable to avoid interactive prompts during package installation.
3. We update the package repository and install the necessary components for a LAMP stack, including Apache, MySQL, and PHP.
4. We enable the Apache mod_rewrite module for URL rewriting.
5. We configure Apache to set the "ServerName" to "localhost."
6. We specify that Apache should run in the foreground as the main process.
7. We expose ports 80 (HTTP) and 3306 (MySQL) for external access.

Dockerfile for Ubuntu-based LEMP(Linux, Nginx, MySQL (or MariaDB), and PHP)

```
# Use the official Ubuntu image as the base
FROM ubuntu:20.04

# Set environment variables to avoid interactive prompts during installation
ENV DEBIAN_FRONTEND=noninteractive

# Update the package repository and install necessary packages
RUN apt-get update && apt-get install -y \
    nginx \
    mysql-server \
    php-fpm \
    php-mysql \
```

```
    php-common \
    php-gd \
    php-cli \
    php-curl \
    php-json \
    php-zip \
    php-mbstring \
    php-xmlrpc \
    php-xml \
    php-bcmath \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# Configure PHP-FPM
RUN sed -i 's/;cgi.fix_pathinfo=1/cgi.fix_pathinfo=0/' /etc/php/7.4/fpm/php.ini

# Start services
CMD service mysql start && service php7.4-fpm start && nginx -g 'daemon off;'

# Expose ports
EXPOSE 80

# Create a directory for the Nginx site configuration
RUN mkdir -p /var/www/html

# Copy a sample Nginx site configuration file
COPY default.conf /etc/nginx/sites-available/default

# Create a symbolic link to enable the site configuration
RUN ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/

# Remove the default Nginx index.html file
RUN rm /var/www/html/index.nginx-debian.html
```

In this Dockerfile:

1. We start with the official Ubuntu 20.04 image as the base.
2. We set an environment variable to avoid interactive prompts during package installation.
3. We update the package repository and install Nginx, MySQL, and PHP packages.
4. We configure PHP-FPM to disable `cgi.fix_pathinfo` for security reasons.
5. We start MySQL, PHP-FPM, and Nginx as the main processes.
6. We expose port 80 for HTTP traffic.
7. We create a directory for the Nginx site configuration and copy a sample Nginx site configuration file.
8. We create a symbolic link to enable the site configuration.
9. We remove the default Nginx index.html file.

You should customize the Nginx site configuration file (`default.conf`) and add your PHP application code to the `/var/www/html` directory.

Dockerfile for a basic Node.js application:

```
# Use an official Node.js runtime as the base image
FROM node:14

# Set the working directory in the container
WORKDIR /app
```

```
6
7  # Copy package.json and package-lock.json to the working directory
8  COPY package*.json ./
9
10 # Install application dependencies
11 RUN npm install
12
13 # Copy the rest of the application source code to the working directory
14 COPY . .
15
16 # Expose a port that the application will listen on
17 EXPOSE 3000
18
19 # Define a command to run the application
20 CMD ["node", "app.js"]
```

In this example:

1. We start with the official Node.js Docker image version 14 as our base image.
2. We set the working directory to `/app` inside the container.
3. We copy `package.json` and `package-lock.json` into the container to install dependencies.
4. We run `npm install` to install the application dependencies.
5. We copy the rest of the application source code into the container.
6. We expose port 3000 for the application to listen on.
7. We specify the command to run when the container starts, which is `node app.js`.