Migrate from Ubuntu

- Key differences
- Migration steps
  - Step 1: Update the base image in your Dockerfile
  - Step 2: Update package installation commands
  - Step 3: Update the runtime image in your Dockerfile
- Language-specific examples

Docker Hardened Images (DHI) come in Alpine-based and Debian-based variants. When migrating from an Ubuntu-based image, you should migrate to the Debian-based DHI variant, as both Ubuntu and Debian share the same package management system (APT) and underlying architecture, making migration straightforward.

This guide helps you migrate from an existing Ubuntu-based image to DHI.

Key differences

When migrating from Ubuntu-based images to DHI Debian, be aware of these key differences:

Item	Ubuntu-based images	Docker Hardened Images
Package management	Varies by image. Some include APT package manager, others don't	Package managers generally only available in images with a `dev` tag. Runtime images don't contain package managers. Use multi-stage builds and copy necessary artifacts from the build stage to the runtime stage.
Non-root user	Varies by image. Some run as root, others as non-root	Runtime variants run as the nonroot user by default. Ensure that necessary files and directories are accessible to the nonroot user.
Multi-stage build	Recommended	Recommended. Use images with a `dev` or `sdk` tags for build stages and non-dev images for runtime.

Item	Ubuntu-based images	Docker Hardened Images
Ports	Can bind to privileged ports (under 1024) when running as root	Run as a nonroot user by default. Applications can't bind to privileged ports (under 1024) when running in Kubernetes or in Docker Engine versions older than 20.10. Configure your application to listen on port 1025 or higher inside the container.
Entry point	Varies by image	May have different entry points than Ubuntu-based images. Inspect entry points and update your Dockerfile if necessary.
Shell	Varies by image. Some include a shell, others don't	Runtime images don't contain a shell. Use `dev` images in build stages to run shell commands and then copy artifacts to the runtime stage.
Package repositories	Uses Ubuntu package repositories	Uses Debian package repositories. Most packages have similar names, but some may differ.

Migration steps

Step 1: Update the base image in your Dockerfile

Update the base image in your application's Dockerfile to a hardened image. This is typically going to be an image tagged as `dev` or `sdk` because it has the tools needed to install packages and dependencies.

The following example diff snippet from a Dockerfile shows the old Ubuntu-based image replaced by the new DHI Debian image.

> You must authenticate to `dhi.io` before you can pull Docker Hardened Images. Use your Docker ID credentials (the same username and password you use for Docker Hub). If you don't have a Docker account, create one for free.
>
> Run `docker login dhi.io` to authenticate.

```
- ## Original Ubuntu-based image
- FROM ubuntu/go:1.22-24.04

+ ## Updated to use hardened Debian-based image
+ FROM dhi.io/golang:1.25-debian13-dev
```

To find the right tag, explore the available tags in the DHI Catalog.

Step 2: Update package installation commands

Since Ubuntu and Debian both use APT for package management, most package installation commands remain similar. However, you need to ensure that package installations only occur in `dev` or `sdk` images, as runtime images don't contain package managers.

```
- ## Ubuntu: Installing packages
- FROM ubuntu/go:1.22-24.04
- RUN apt-get update && apt-get install -y \
-     git \
-     && rm -rf /var/lib/apt/lists/*

+ ## DHI: Use a language-specific dev image with package manager
+ FROM dhi.io/golang:1.25-debian13-dev
+ RUN apt-get update && apt-get install -y \
+     git \
+     && rm -rf /var/lib/apt/lists/*
```

Most Ubuntu packages are available in Debian with the same names. If you encounter missing packages, you can search for equivalent packages using the Debian package search website.

Step 3: Update the runtime image in your Dockerfile

> Multi-stage builds are recommended to keep your final image minimal and secure. Single-stage builds are supported, but they include the full `dev` image and therefore result in a larger image with a broader attack surface.

To ensure that your final image is as minimal as possible, you should use a multi-stage build. All stages in your Dockerfile should use a hardened image. While intermediary stages will typically use images tagged as `dev` or `sdk`, your final runtime stage should use a runtime image.

Utilize the build stage to install dependencies and prepare your application, then copy the resulting artifacts to the final runtime stage. This ensures that your final image is minimal and secure.

The following example shows a multi-stage Dockerfile migrating from Ubuntu to DHI Debian:

```
# Build stage
FROM dhi.io/golang:1.25-debian13-dev AS builder
WORKDIR /app

# Install system dependencies (only available in dev images)
RUN apt-get update && apt-get install -y \
    git \
    && rm -rf /var/lib/apt/lists/*

# Copy application files
COPY go.mod go.sum ./
RUN go mod download

COPY . .
RUN CGO_ENABLED=0 GOOS=linux go build -a -ldflags="-s -w" -o main .

# Runtime stage
FROM dhi.io/golang:1.25-debian13
WORKDIR /app

# Copy compiled binary from builder
COPY --from=builder /app/main /app/main

# Run the application
ENTRYPOINT ["/app/main"]
```

Language-specific examples

See the examples section for language-specific migration examples:

- Go

- Python
- Node.js

□□□ #2
□ □□□ □□□ 2026-10-31 19:18:39 UTC
□ □□□ □□□ 2026-10-31 19:18:40 UTC