Build and share a containerized application

- Before you start
- Run a container
- Run an application stack
- Build an image
- Share the image
- Clean up
- What you learned
- What's next

An application can depend on a particular runtime, libraries, database, and supporting tools. Docker packages these dependencies with the application so you can run it consistently without preparing each machine by hand.

In this 15-minute tutorial, you'll run a container, start a multi-container application, package the application as an image, and share the image through Docker Hub.

Before you start

- Install Docker Desktop and start it
- Install Git
- Create a Docker account

Run a container

Start with a small web application that someone else has packaged for Docker. Docker distributes applications in images. An image is a ready-to-run package that contains an application and

everything it needs. A container is a running instance of an image.

Download the image from Docker Hub:

```
$ docker pull docker/welcome-to-docker
```

Start a container from the image:

```
$ docker run --detach --name welcome --publish 8080:80 docker/welcome-to-docker
```

This command runs the container in the background (`--detach`), names it `welcome` (`--name`), and makes its web server available at port 8080 on your machine (`--publish 8080:80`). The final argument identifies the image to run.

Open http://localhost:8080 to see the application. You downloaded an image and started one container from it.

Remove the container before continuing:

```
$ docker rm --force welcome
```

This removes the running application. The image remains available locally, so Docker can create another container from it later.

Run an application stack

The first application needed only one container. Applications often have several parts, such as a frontend, an API, and a database. You could start each part with a separate `docker run` command, but you would also need to keep their configuration and connections in sync.

Docker Compose describes all the parts of an application in a `compose.yaml` file and manages them together. Compose is included with Docker Desktop.

Try it with a prepared to-do application. Clone the project and open its directory:

```
$ git clone https://github.com/docker/getting-started-todo-app
$ cd getting-started-todo-app
```

The project's `compose.yaml` file defines five services: a frontend, an API, a database, a database management interface, and a proxy. A service represents one part of the application and runs in its own container.

Start the complete development stack:

```
$ docker compose up --build --detach
```

The `docker compose up` command reads `compose.yaml` and starts its services. The `--build` option builds the frontend and API images from the project, and `--detach` leaves the containers running in the background. Compose also pulls the images for the supporting services and connects the containers.

Open http://localhost, then add an item to the to-do list.

See the containers that Compose started:

```
$ docker compose ps
```

Each row represents a container for one of the application's services. Instead of preserving a collection of commands, the project keeps the complete stack in one version-controlled definition.

Build an image

So far, you have run a published image and used Compose to build and manage a development stack. Next, build an image of your own that packages the to-do application's frontend and API together.

The repository contains a `Dockerfile`, which is a set of instructions for building the image. Replace `<YOUR_DOCKER_USERNAME>` with your Docker username, then run:

```
$ docker build --tag <YOUR_DOCKER_USERNAME>/getting-started-todo-app .
```

The `--tag` option gives the image a name. The username prefix identifies where the image will be stored on Docker Hub. The final `.` tells Docker to find the `Dockerfile` and application source in the current directory.

Create a container from your image, as you did with the welcome image:

```
$ docker run --detach --name todo --publish 8080:3000 <YOUR_DOCKER_USERNAME>/getting-started-todo-app
```

Open http://localhost:8080. This time, the frontend and API are running together from the image you built.

Share the image

Images can be shared through a registry. Docker Hub is a registry for storing images and making them available to other machines and deployment systems.

In Docker Home, create a public repository named `getting-started-todo-app` under your Docker username.

Sign in from the command line, then push the image:

```
$ docker login
$ docker push <YOUR_DOCKER_USERNAME>/getting-started-todo-app
```

The `docker push` command uploads the image to Docker Hub. Open the repository in Docker Home to see it. Another machine or deployment system can pull and run the same image.

Clean up

Remove the standalone container and the development stack:

```
$ docker rm --force todo
$ docker compose down --volumes
```

What you learned

You ran software packaged by someone else, started a multi-container application with Compose, built your own image, and published it. An image packages an application, a container runs that image, Compose coordinates multiple containers, and a registry makes images available beyond one machine.

What's next

Explore the concepts from this tutorial in more detail:

- What is a container?
- What is an image?
- What is Docker Compose?
- What is a registry?

Choose a language-specific Docker guide to containerize an application of your own.

□□□ #2
□ □□□ □□□ 2026-10-31 18:44:17 UTC
□ □□□ □□□ 2026-10-31 18:44:18 UTC