How to Export and Import Docker Containers and images

Last Updated : 21 Jun, 2024

In software development, flexibility and portability are important. Suppose you've built a perfect application in a Docker container on your development machine and now you need to move this same setup to a colleague's machine or a production server. How do you ensure your application, and all its dependencies and config, moves without a hitch? The answer is in exporting and importing Docker containers and images.

This post will walk you through the steps to export and import Docker containers and images so you can move your work across environments. Whether you're sharing with a team, moving to a new server, or just backing up your containers, you must master these techniques. Let's get started and learn how to make your Dockerized applications as portable as possible.

Key Terminologies

Docker

Docker is an open-source platform designed to automate the deployment of applications as lightweight, portable containers that run virtually anywhere. Docker is a platform that uses containerization to streamline software development.

Docker Image

A Docker image is a read-only template used to create containers. It is a lightweight, executable, stand-alone package that includes everything needed to run the software, including the code, a runtime, libraries, environment variables, and config files.

Container

A container is a runtime instance of an image. It provides a way to run an application in an isolated environment, encapsulating all the necessary dependencies and libraries, and eliminating potential conflicts between applications.

Dockerfile

A Dockerfile is a simple text file that contains a set of commands that Docker uses sequentially to build an image. To read more about Dockerfile, you can read this article.

Exporting and Importing Docker Images

Step-by-Step Process to Export a Docker Image

1. **Identify the Image ID or Name:** To export an image, you need its ID or name. List all images using:

```
docker images
```

2. **Save the Image:** Use the **docker save** command to save the image to a tarball.

```
docker save -o <filename>.tar <image_name>
```

3. **Verify the Saved Image:** Ensure that the tarball has been created successfully in the specified location.

Step-by-Step Process to Import a Docker Image

1. **Transfer the Tarball:** Move the tarball to the target system where you want to import the image.

2. **Load the Image:** Use the **docker load** command to load the image from the tarball.

```
docker load -i <filename>.tar
```

3. **Verify the Loaded Image:** List all images to verify the image has been loaded.

```
docker images
```

Exporting and Importing Docker Containers

Step by Step Process to Export a Docker Container

1. **Identify the Container ID or Name:** To export a container, you need its ID or name. You can list all running containers using the following command:

```
docker ps
```

2. **Export the Container:** Use the **docker export** command to export the container to a tarball.

```
docker export <container_id> -o <filename>.tar
```

3. **Verify the Exported File:** Ensure that the tarball has been created successfully in the specified location.

Step by Step Process to Import a Docker Container

1. **Transfer the Tarball:** Move the tarball to the target system where you want to import the container.

2. **Import the Container:** Use the **docker import** command to import the tarball into Docker as an image.

```
docker import <filename>.tar <new_image_name>
```

> ***Note:*** *Importing a Docker container involves converting the exported tarball of a container's filesystem back into a Docker image instead. This process does not directly create a new running container but provides an image from which you can launch new containers.*

3. **Run a New Container:** Create and run a new container from the imported image.

Example: Exporting and Importing Docker Images and Containers

Project Setup

Let's setup a basic express server to illustrate the whole process from spinning up a container to exporting it:

1. Create a **package.json** file to manage the project dependencies.

File: package.json

```
{
  "name": "basic-express-server",
  "version": "1.0.0",
  "description": "Demo for exporting and importing docker containers/images",
  "main": "./src/app.js",
  "scripts": {
    "start": "node ./src/app.js"
  },
  "author": "Your Name",
  "license": "ISC",
  "dependencies": {
    "express": "^4.19.2"
  }
}
```

Run **npm install** to install the dependencies.

2. Create a file named **app.js** in **src** folder.

File: src/app.js

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```

3. Let's create a **Dockerfile** to containerize the Express server.

File: Dockerfile

```
FROM node

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

CMD [ "npm", "start" ]

EXPOSE 3000
```

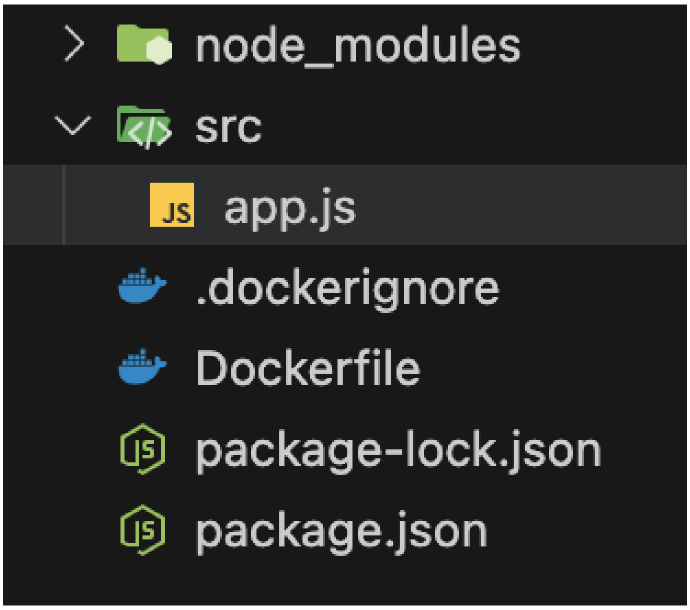
4. Create a **.dockerignore** file to prevent copying unnecessary files to the Docker image.

File: .dockerignore

```
node_modules
npm-debug.log
```

Folder Structure

This is how your folder structure should look like after following the above steps:



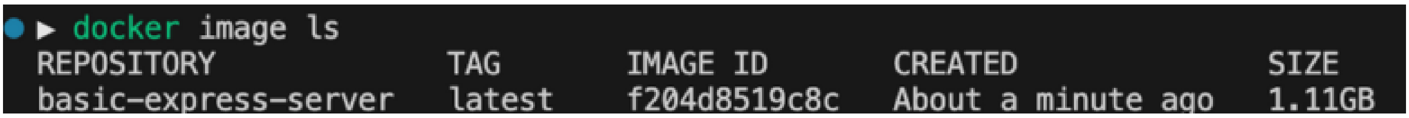
```
> node_modules
v src
    app.js
  .dockerignore
  Dockerfile
  package-lock.json
  package.json
```

5. Let's build an image of our Express server by running this command:

```
docker build -t basic-express-server .
```

6. Verify the created image by listing all the images:

```
docker image ls
```



```
● ▶ docker image ls
REPOSITORY             TAG       IMAGE ID       CREATED              SIZE
basic-express-server   latest    f204d8519c8c   About a minute ago   1.11GB
```

Exporting Docker Image

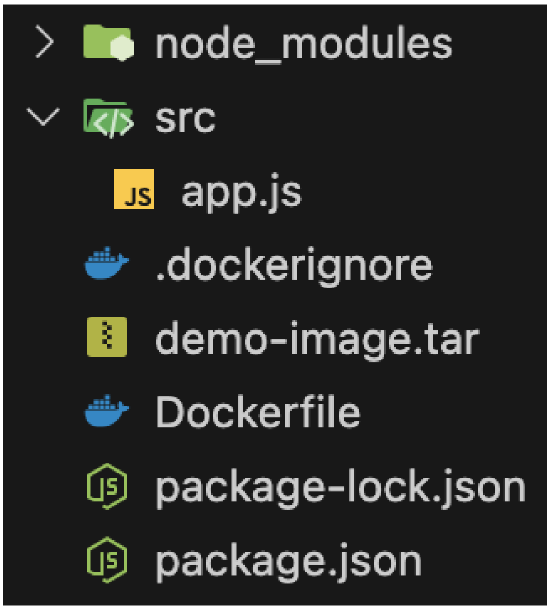
1. Save the image to a tarball:

```
docker save -o demo-image.tar basic-express-server
```

where,

-o, --output string Write to a file, instead of STDOUT

Check if the tarball file is generated or not.



```
> node_modules
v src
    app.js
  .dockerignore
  demo-image.tar
  Dockerfile
  package-lock.json
  package.json
```

2. Let's first our already built image so that we can load the image again using the tarball file. You can remove the image using this command:

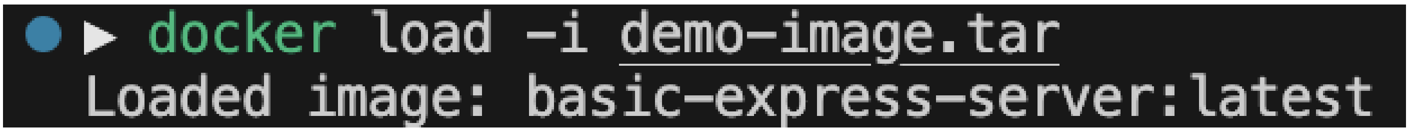
```
docker rmi <image_id>
```

Check whether the image is deleted or not by listing all the images.

Importing Docker Image

3. Now, let's load the image again using the tarball file:

```
docker load -i demo-image.tar
```



```
▶ docker load -i demo-image.tar
  Loaded image: basic-express-server:latest
```

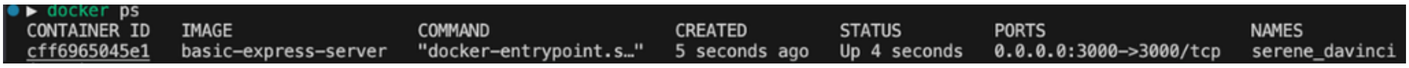
Again, list all the images to see the newly loaded image.

4. Let's now run a container using this image:

```
docker run -p 3000:3000 -d basic-express-server
```

You can list down the running containers using:

```
docker ps
```



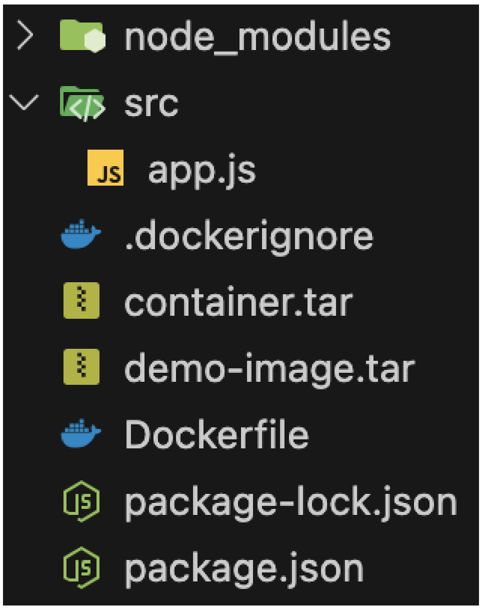
```
▶ docker ps
  CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS                    NAMES
  cff6965045e1   basic-express-server   "docker-entrypoint.s…"   5 seconds ago   Up 4 seconds   0.0.0.0:3000->3000/tcp   serene_davinci
```

Exporting Docker Container

5. To export this container, run:

```
docker export cff6965045e1 -o container.tar
```

A tarball file will be created named **container.tar:**

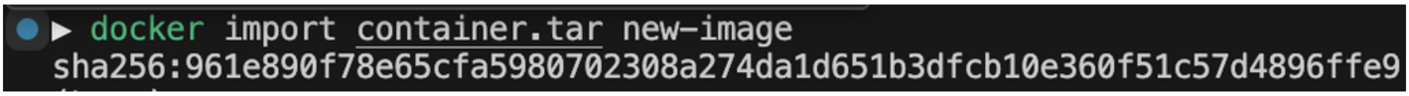


```
> node_modules
v src
    app.js
  .dockerignore
  container.tar
  demo-image.tar
  Dockerfile
  package-lock.json
  package.json
```

Importing Docker Container

6. Import the Docker container by running:

```
docker import container.tar new-image
```



```
● ▶ docker import container.tar new-image
  sha256:961e890f78e65cfa5980702308a274da1d651b3dfcb10e360f51c57d4896ffe9
```

List all the images and verify if the new image is present there or not.

7. Run a container using the new image created by importing the tarball file and check if everything is working fine.

Exporting and Importing Docker Images/Containers - FAQs

What is the difference between `docker export` and `docker save`?

> *`docker export` saves the filesystem of a container to a tarball, while `docker save` saves a Docker image to a tarball. `docker export` does not include image layers and metadata, whereas `docker save` preserves the image layers and metadata.*

Can I export a stopped container?

Yes, you can export both running and stopped containers using the `docker export` command.

How do I transfer a Docker image between different hosts?

You can use `docker save` to save the image to a tarball, transfer the tarball to the target host, and then use `docker load` to load the image on the new host.

Is there a way to export multiple containers at once?

You would need to export each container individually using `docker export`. There isn't a built-in command to export multiple containers simultaneously.

What happens to my data if I export and import a container?

The exported container's filesystem, including all data within it, is preserved in the tarball. When you import it, the container's state is restored, including the data.

Comment More info

Next Article >

Similar Reads

How to Export and Import Docker Containers and images

In software development, flexibility and portability are important. Suppose you've built a perfect application in a Docker container on your development machine and now you need to move this same...

6 min read

How to Push a Container Image to a Docker Repository?

In this article we will look into how you can push a container image to a Docker Repo. We're going to use Docker Hub as a container registry, that we're going to push our Docker image to. Follow the below step...

🕓 2 min read

Remove All Containers and Images in Docker

Pre-requisite: Docker In Docker , if we have exited a container without stopping it, we then need to stop them manually as it has not stopped on exit. Similarly, for images, we need to delete them from top to...

🕓 4 min read

How To Create a Docker Container from an Existing Image?

Docker is an open-source software, that is used to contanerize our applications. Containerizing applications makes deployment a lot easier. For containerizing applications, docker uses Docker images,...

🕓 10 min read

How to Use Ansible for Docker Container Management

Containerization has become one of the foundations for realizing scalable, reliable, and efficient application deployments in modern DevOps practice. Docker as a leading containerization platform...

🕓 7 min read

How to Run MongoDB as a Docker Container?

MongoDB is an open-source document-oriented database designed to store a large scale of data and allows you to work with that data very efficiently. It is categorized under the NoSQL (Not only SQL)...

🕓 4 min read

How to Remove Docker Containers

Docker is a platform that provides a set of PaaS (Platform as a Service) products that help developers containerize applications, allowing them to run consistently across various supported environments. The...

🕓 2 min read

How To Push A Docker Image To Amazon ECR?

We go over how to submit a Docker image to the Amazon Elastic Container Registry (ECR) in this tutorial. By offering a safe, scalable registry for storing and distributing Docker images inside the AWS ecosyste...

🕓 4 min read

How To See Docker Image Contents?

In the evolving world of containerization, Docker has emerged as a tool for packaging and deploying applications. While creating and sharing Docker images has streamlined the development and...

🕓 3 min read

Connecting Two Docker Containers Over the Same Network

Whenever we expose a container's port in docker, it creates a network path from the outside of that machine, through the networking layer, and enters that container. In this way, other containers can...

🕓 4 min read

How Can I Expose More Than 1 Port With Docker ?

Docker is an important tool that simplifies the deployment process by encapsulating applications within a container. To facilitate various types of communication in a Docker container, we need to expose more...

🕓 4 min read

How to Use Docker Images For Scaling Applications?

Docker image is a lightweight piece of software that includes everything that your application needs to run, it includes code, software packages, and libraries. Docker images are built using a Docker file which...

🕓 4 min read

How to Link Multiple Container in Docker ?

Docker is a free software created by Docker Inc. and it enables clients to create free and confined environments where the users can deploy their applications. A container is a standard unit of software...

🕓 2 min read

Copying Files to and from Docker Containers

While working on a Docker project, you might require copying files to and from Docker Containers and your Local Machine. Once you have built the Docker Image with a particular Docker build context,...

🕓 9 min read

How to Continue a Docker Container Which has Exited

Docker containers have reformed how programming is developed, deployed, and managed, offering unmatched productivity and consistency across various conditions. In any case, overseeing containers,...

🕓 6 min read

How To Build Docker Image In GitHub Actions ?

GitHub Actions is a powerful tool provided by GitHub to automate tasks directly within your repositories. It plays a crucial role in Continuous Integration/Continuous Deployment (CI/CD) pipelines, enabling...

🕓 5 min read

How Do I Edit A File After I Shell To A Docker Container?

In containerization, the `docker exec` command stands out as a powerful tool for interacting with running Docker containers. This article explores the capabilities and usage of `docker exec` and edits the files in...

4 min read

How To Use Docker Security Tools To Secure Docker Container Images

Docker is a tool that is used to package the application and its dependencies into compact units called Docker containers. In this guide, I will first discuss what a docker is. Then I will discuss a security tool...

11 min read

How to Setup Jenkins in Docker Container?

Setting up of Jenkins in a docker container provides the facility of streamlining the CI/CD with scalability and consistency. It also helps for attaching the worker nodes as slaves and run the build jobs over their. I...

6 min read

Article Tags : Docker DevOps