Docker best practices you need to know

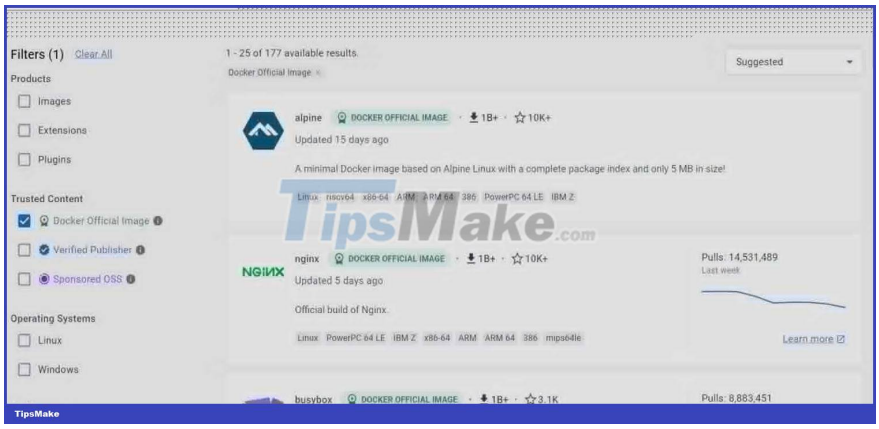
Docker is the most popular containerization software, but not everyone knows how to use it effectively. The tips below will help you maximize the power of Docker.

Use official Docker Image

When containerizing your application, you must use a Docker image. You can build an image with custom configuration or use the official Docker image.

Building your own image requires you to handle all the configuration yourself. For example, to build an image for a node.js application, you must download node.js and its dependencies. This process is time consuming and may not yield all the correct configurations.

Docker recommends using the official node.js image, with all the correct dependencies. Image Docker has better security metrics, is lightweight, and has been tested for a variety of environments. You can find the official image on Docker's official site.



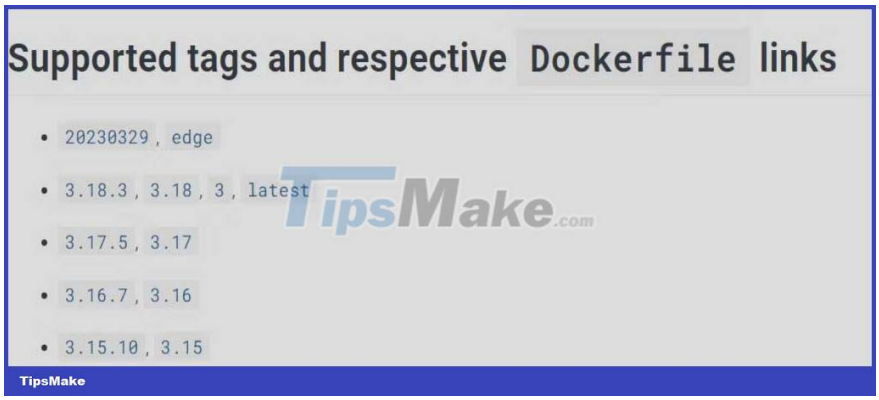
Use specific versions of Docker Images

Usually, when you download an official image, it comes with the latest tag representing the highest version of that image. Each time you build a container from that image, it is a different version than the closest container.

Building with other Docker image versions can easily expose you to unpredictable behavior in your application. Versions can conflict with other dependencies, even causing the app to crash.

Docker recommends that you download and build with images of a specific version. The official images are also documented and cover the most common use cases.

For example, instead of docker pull alpine, use **docker pull alpine:3.18.3** . Docker will get that specific version. You can then use it in successive builds, reducing errors in your app. You can find specific versions of the image on the official Docker image page, under **Supported tags and respective Dockerfile links** :



Scan images to find security holes

How do you know an image you want to build is free of security vulnerabilities? By scanning it. You can scan Docker images with the scan docker command as follows:

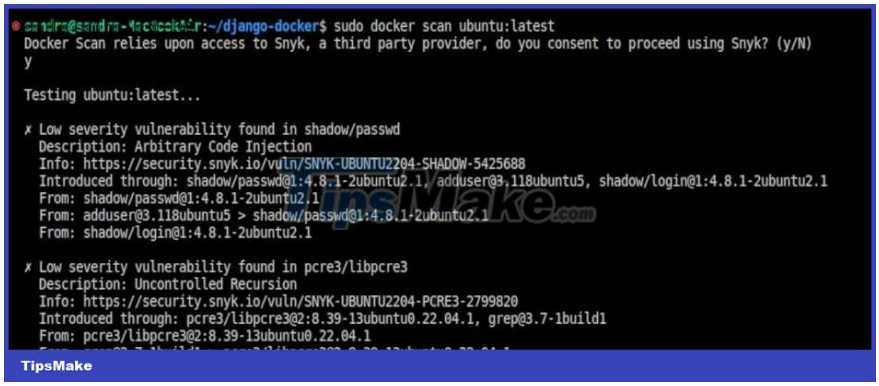
```
docker scan [IMAGE]
```

First, you must log in to docker to scan an image:

```
docker login
```

Then, scan the specific image you want to test:

```
docker scan ubuntu:latest
```



A tool called Synk scans this image, listing any vulnerabilities by severity. You can see the vulnerability type and links to information about it, including how to fix it. From the scanning process, you can know whether the

image is secure enough for the application or not.

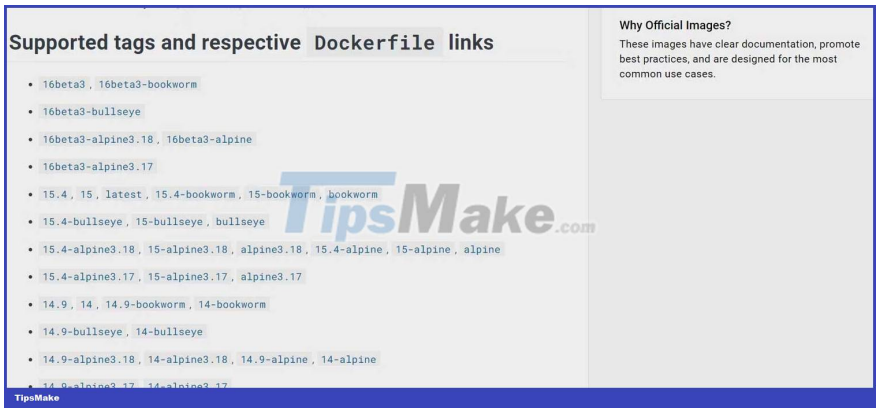
Use small Docker Image

When you download a Docker image, it comes with all the system utilities. This increases image size with tools you don't need.

Large Docker images 'eat up' storage space and can slow down container runtime. They are also more likely to suffer from more security vulnerabilities.

You can reduce the size of Docker images using Alpine images. Alpine images are lightweight and contain only the necessary tools. They reduce storage space, making applications run faster and more efficiently.

You will find an Alpine version for most official images on Docker. Here is an example of the Alpine version for PostgreSQL:

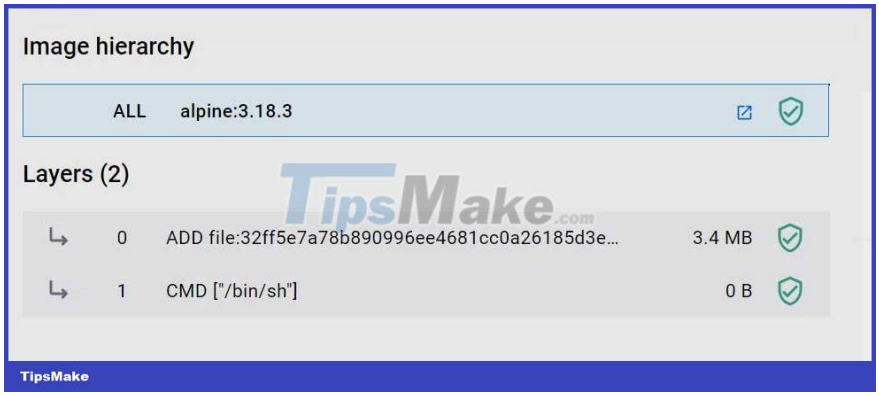


Optimize the storage image layer

Each command in the Dockerfile represents a layer on the image. These layers have different utilities and functions. If you look at the official images on Docker Hub, you will see the instructions used to create them.

The Dockerfile includes everything you need to create the image. It is one of the reasons why many programmers prefer Docker over virtual machines.

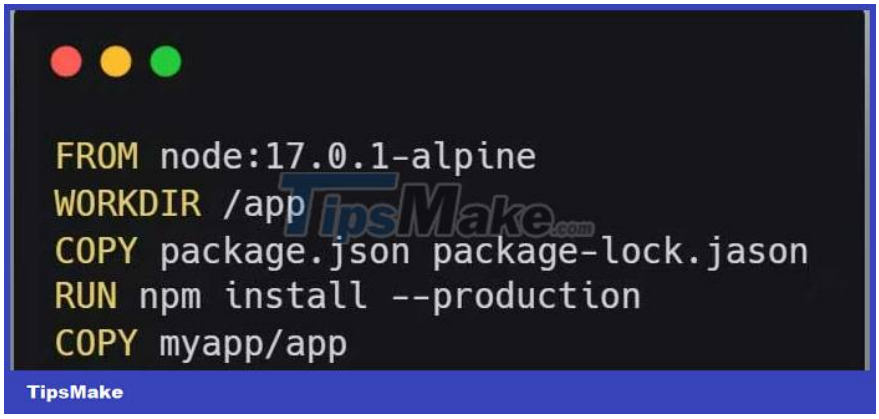
Here is the structure of an example Alpine image:



When building an image-based app, you are adding more layers to that image. Docker runs instructions on the Dockerfile from top to bottom, and if one layer changes, Docker must rebuild the next layers.

Best practice for organizing Dockerfiles from files that change least to most often. Immutable instructions, such as installation, may be at the top of the file.

When you change files, Docker builds from the changed files and stores the unchanged files above it. Therefore, the process runs faster.



Take a look at the example illustrated in the image above. If there are changes in the application files, Docker will build from there. It doesn't have to reinstall the npm package.

If building from this image, the process will run faster than rebuilding all the other layers from scratch. Storage also speeds up image loading and pushing from Docker Hub.

Above are the best tips for using Docker. Hope the article is useful to you.

You finished reading the article "**Docker best practices you need to know**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.