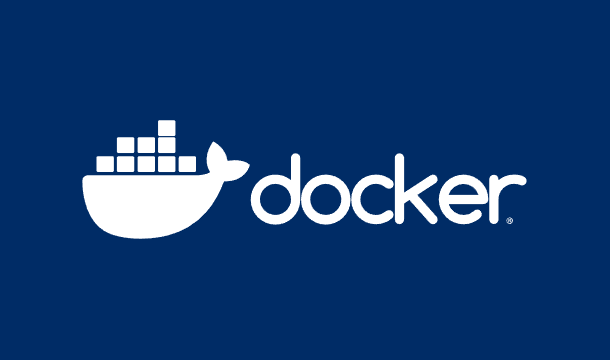
**Blog :** Create a Docker Image from a Running Container

What we're going to do is deploy a container, for an NGINX server, modify it, and then create a new image from that running container that you can then use to base new containers from.



We will start with basic containers in Docker that makes learning and working with container technology will be easy.

Lets continue with our first step with Docker. I’m going to walk you through some of the first things you might do with Docker containers. What we’re going to do is deploy a container, for an [NGINX server](https://www.nginx.com/?utm_content=inline+mention), modify it, and then create a new image from that running container that you can then use to base new containers from.

Why would you do this? Say, for example, you do use NGINX for most of your web-based container deployments. Instead of having to always deploy a new NGINX container and then modify it to meet your baseline needs, you can create a single image that already contains that baseline and avoid a lot of repetitious work.

It’s all in the name of efficiency.

With that said, let’s get to work.

**Install Docker**

On the off-chance you don’t already have Docker installed, let’s do so. I’ll be demonstrating on my go-to Ubuntu Server (version 20.04). If you use a different distribution of Linux for your container deployments, you’ll need to only modify the installation steps for this process.

To install Docker on your Ubuntu Server, log in and issue the following command:

**TRENDING STORIES**

1. [**Apple Containers on macOS: A Technical Comparison With Docker**](https://thenewstack.io/apple-containers-on-macos-a-technical-comparison-with-docker/)
2. [**Containerized Apps for Your Home Network**](https://thenewstack.io/containerized-apps-for-your-home-network/)
3. [**Docker Basics: How to Use Dockerfiles**](https://thenewstack.io/docker-basics-how-to-use-dockerfiles/)
4. [**Build Your Own Private Cloud at Home With Docker**](https://thenewstack.io/build-your-own-private-cloud-at-home-with-docker/)
5. [**How To Build Scalable and Reliable CI/CD Pipelines With Kubernetes**](https://thenewstack.io/how-to-build-scalable-and-reliable-ci-cd-pipelines-with-kubernetes/)

**sudo apt-get install docker.io -y**

After the installation completes, add your user to the docker group with:

**sudo usermod -aG docker $USER**

Log and out log back in so the changes take effect.

**Create the New Container**

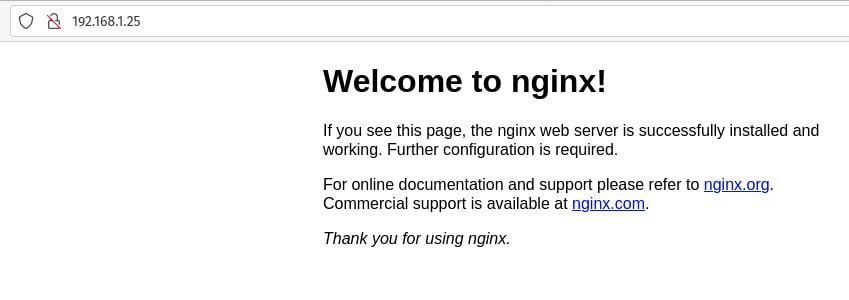
With Docker ready, let’s create the new container. This will be a very basic web server, using NGINX. The command looks like this:

**docker create --name nginx-base -p 80:80 nginx:alpine**

For those who’ve never worked with Docker, that command does the following:

Creates a new container with the name **nginx-base**, that runs on internal (guest) port 80 and external (host) port 80, and uses the **nginx:alpine** image that it will have to pull down from DockerHub.

The command will respond with the container ID, which is a long string of random characters, indicating the deployment was successful. If you open a web browser and point it to the IP address of the hosting server, you should see the NGINX welcome page (**Figure A**).



***Figure A:****The NGINX welcome page, as produced by our Docker container.*

**Modify the Existing Container**

Now, it’s time to modify our existing container. We’re only going to do a very basic modification (which you can later expand on to your heart’s content). What we’re going to do is create a new index.html page for NGINX to serve.

To do this, let’s create the new page with the command:

**nano index.html**

In that file, paste the following contents (you can modify it to say whatever you want):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | &lt;html&gt;  &lt;head&gt;  &lt;title&gt;Hello, New Stack!&lt;/title&gt;  &lt;/head&gt;  &lt;body&gt;  &lt;h1&gt;Hello, New Stack!&lt;/1&gt;  &lt;/body&gt; |

Save and close the file.

As you can see, the new welcome page will say “Hello, New Stack!*“* Anyone that’s created a “Hello, World” application will recognize that immediately.

Okay, for our next trick, we’re going to copy that file to the running container. With the NGINX image, the document root (the base directory that houses web pages) is */usr/share/nginx/html*. And because Docker has a built-in copy command, adding the file to the running container is very simple.

Remember, our container is called nginx-base. The command to copy index.html to the document root on nginx-base is:

*docker cp index.html nginx-base:/usr/share/nginx/html/index.html*

If you refresh the page in your web browser, you should see it now displays the new welcome message (**Figure B**).



***Figure B****: Our new welcome message within our running Docker container.*

**Create a New Image—++++**

Alrighty then, we have the newly modified (and running) container. How do we create a new image that includes the changes? It’s actually very simple.

First, we commit the changes with the command:

**docker commit nginx-base**

What this will do is create a new image without a repository or tag. List out the current images with the command:

**docker images**

What you should see is something like this:

|  |  |
| --- | --- |
| 1  2  3 | REPOSITORY   TAG       IMAGE ID          CREATED               SIZE  &lt;none&gt;       &lt;none&gt;    7fb3d1656c23   About a minute ago       23.5MB  nginx        alpine    cc44224bfe20   7 days ago               23.5MB |

The bottom image is the one we used to create our new container. The top image is the one we just created. Notice that it doesn’t have either a REPOSITORY (the first column) or a TAG (the second column). For this image to be usable, we have to tag it. In order to tag the image, we have to use the IMAGE ID as an identifier, so tag the image (we’ll name it docker-base-image) like this:

**docker tag IMAGE\_ID nginx-base-container**

Where IMAGE\_ID is the actual ID of your new container.

Now, if you list the images (using the *docker images* command), you’ll see something like this:

|  |  |
| --- | --- |
| 1  2  3 | REPOSITORY               TAG       IMAGE ID          CREATED         SIZE  nginx-base-container     latest    7fb3d1656c23   6 minutes ago      23.5MB  nginx                    alpine    cc44224bfe20   7 days ago         23.5MB |

There you go, you’ve created a new Docker image from a running container. Let’s stop and remove the original container. To do that, locate the ID of the original with the command:

**docker ps -a**

With the first 4 characters of the original container ID, stop it with:

**docker stop ID**

Where ID is the first four digits of the original container.

Remove the container with the command:

**docker rm ID**

Where ID is the first four digits of the original container.

You could then deploy a container from the new image with a command like:

**docker create --name nginx-new -p 80:80 nginx-base-container:latest**

Refresh your web browser and you should, once again, see the new Hello, New Stack! welcome page. The new image includes our modifications to the index.html page, so every container created from it will reflect that change.

And that’s how easy it is to create a Docker image from a running container. This skill will come in very handy (especially as you build on it as you go).

Blog: Gen AI

**Generative AI & DevOps: Beginner Tutorial and Integration Guide**

**What is Generative AI?**

Generative AI (Gen AI) refers to AI systems that can create new content—text, images, code, or infrastructure definitions—using trained models and machine learning algorithms. Popular Gen AI solutions include GPT (OpenAI), Stable Diffusion, and specialized LLMs for code and operations. Gen AI uses deep learning to automate the creation and innovation process, making it a powerful tool in the technology ecosystem.

**Why Should DevOps Engineers Learn Gen AI?**

* Automates repetitive tasks: Speeds up code review, incident detection, and infrastructure automation.
* Boosts reliability and innovation: Predicts failures, generates optimal CI/CD pipeline configs, and writes Infrastructure-as-Code (IaC).
* Frees up engineering time: Lets teams focus on problem-solving and innovation instead of manual operations.

**Getting Started with Gen AI**

1. Learn Gen AI Basics
   * Study deep learning, transformer models, and prompt engineering.
   * Use tutorials from W3Schools, Microsoft, or YouTube.
2. Choose a Framework
   * Try TensorFlow, PyTorch, Hugging Face, or LangChain.
   * Experiment with pre-trained models before building custom ones.
3. Set Up Your Dev Environment
   * Use Python and Jupyter notebooks, or set up cloud AI services (Azure, AWS, Hugging Face).
4. Work on a Small Project
   * Automate documentation, create a Q&A bot, or generate code snippets using LLMs for immediate value.

**Starter Gen AI Projects for Beginners & DevOps**

**General AI Projects**

* Poem Generator: Enter a topic, output a poem.
* Image-to-Story: Upload an image, get a creative story.
* Affirmation Bot: Daily positive messages using Gen AI.
* Song Lyrics Maker: Text generation based on genre.
* Recipe Generator: Gen AI creates recipes from ingredients.

**DevOps-Focused Projects**

* Script Generator: Automatically generate Bash/Python scripts for infra tasks.
* Pipeline-as-Code Creator: Build a Gen AI tool that creates YAML configs for CI/CD tools.
* Test Data Generator: Generate realistic and edge case test data for automated testing.
* Issue Summarizer Bot: Summarizes logs or incidents, surfaces root causes and actions.
* Gen AI-enhanced CI Workflow: Integrate Gen AI into CI/CD for code analysis and recommendations.

**DevOps Tasks That Benefit Most from Gen AI Automation**

* Code Generation & Debugging: Automates script writing and bug detection for deployment and operations.
* CI/CD Pipeline Optimization: Auto-generates pipeline configs, detects bottlenecks, self-healing actions.
* Automated Testing: Builds test cases, enhances coverage, prioritizes high-risk code.
* Monitoring & Incident Response: Detects anomalies, generates alerts, recommends fixes.
* Infrastructure as Code (IaC): Creates and optimizes cloud configs (Terraform, Kubernetes).
* Documentation Creation: Updates references, resources, and guides from codebase changes.
* Resource/Security Management: Predicts scaling demands, optimizes cloud costs, enhances security scans.