docker pull alpine:latest

docker run alpine

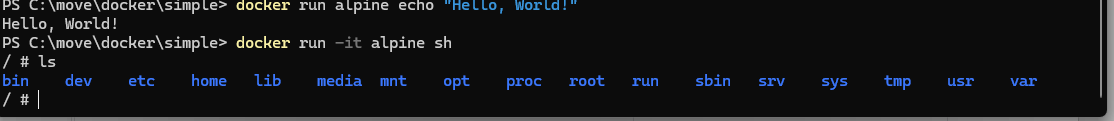
docker run alpine echo "Hello, World!"

keep iteractive:

docker run -it alpine sh

/ # ls

/ # exit



<https://www.sitepoint.com/node-js-docker>

1. Execute code in a given image

Run the below to execute the version.js file:

docker run --rm --name version -v ${PWD}:/home/node/app -w /home/node/app node:lts-alpine version.js

execute, remove after the execution is terminated, create image with name ‘version’, mount current working directory to /app, cd in app, from the image alpine, file to executed.

1. Build image to be reused

Create the related Dockerfile image file under the ‘simple’ folder, the execute the following to create your image:

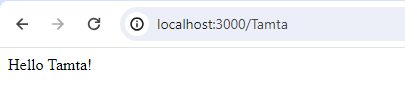
docker image build -t simple .

Start a container with port 3000 on host port 3000 from my created image:

docker run -it --rm --name simple -p 3000:3000 simple

Try to access the on the browser:

<http://localhost:3000/Tamta>



To stop the container

docker container stop simple

**A Better Docker Development Workflow**

Create a new file named docker-compse.yml with the needed content then

Start your app running in debug mode with:

$ docker compose up

### To Stop the container

docker compose down

# Ease of Scaling

You can scale services with just one command. For example, scaling the app service to 3 instances:

bash

$ docker-compose up --scale app=3

# Logs and Monitoring

View logs for all services in one place:

docker-compose logs

# Tail logs dynamically:

docker-compose logs -f

# Debug with VS Code

Open the VS Code **Run and Debug** panel and click **create a launch.json file**.

Choose **Node.js** in the dropdown and a .vscode/launch.json file is created and opened in the editor. Add the settings code which attaches the debugger to the running container

Save the file then click **Attach to Container** at the top of the Debug pane to start debugging.