TECH TIP — BEGINNER/INTERMEDIATE

What Docker Is and Why "It Runs in a Container" Matters

The thing everyone says and nobody explains

From Nolan Law Firm — nemolegal.com/tools

The Actual Problem Docker Solves

"It works on my machine" is one of the most expensive phrases in software. You build something, it runs perfectly, you put it on a server, it breaks. Different Python version. Missing library. Wrong system settings. Docker's entire purpose is to make that problem go away.

What a Container Is

A container is a packaged, isolated environment that includes your app and everything it needs to run — the runtime, the libraries, the config. It runs the same way on your laptop, your server, or anyone else's machine. The container is sealed. It doesn't care what's installed on the host system.

> **Analogy:** A shipping container holds cargo. The ship doesn't care what's inside. The port doesn't care what ship it's on. Docker containers work the same way — the app is packaged, isolated, and portable.

Container vs. Virtual Machine

	Virtual Machine	Docker Container
Boots	Full OS — minutes	Your app only — seconds
Size	Gigabytes	Megabytes to hundreds of MB
Isolation	Complete separate OS	Shares host OS kernel
Use case	Run a totally different OS	Ship an app consistently
Overhead	Heavy	Light

The Commands You'll Actually Use

```
docker ps # list running containers

docker ps -a # list all containers including stopped
```

```
docker logs container-name # see output/errors from a container

docker exec -it container-name bash # open a shell inside a container

docker stop container-name # stop a running container

docker restart container-name # restart it

docker compose up -d # start all services defined in docker-compose.yml

docker compose down # stop and remove them
```

The docker-compose.yml File

Most real setups use Docker Compose — a file that defines all your containers and how they connect. Instead of running a long docker command with 15 flags, you write it once in the file, then just run `docker compose up -d`. The `-d` means detached — it runs in the background.

```
# Minimal docker-compose.yml example:

services:

myapp:

image: nginx:latest

ports:

- "8080:80"

volumes:

- ./html:/usr/share/nginx/html
```

When Something Breaks in a Container

The first command is always `docker logs container-name`. If that's not enough, `docker exec -it container-name bash` puts you inside the container so you can poke around exactly like you would on a regular server.

Using AI to Help With This You don't have to fully understand this to use it. Here are prompts that work:

```
"I need to run [app/service] in Docker. Write me a docker-compose.yml that sets it up with [specific requirements]."
```

```
"My Docker container is crashing. Here are the logs: [paste]. What is wrong and how do I fix it?"
```

```
"Explain what this docker-compose.yml is doing and whether there are any problems with it: [paste]."
```
