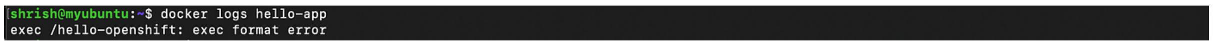
Running x86 (Linux) Containers on ARM (macOS) with QEMU

Over the past couple of days, I was trying to run the classic `openshift/hello-openshift` container image locally on my VM… and kept hitting this frustrating error:

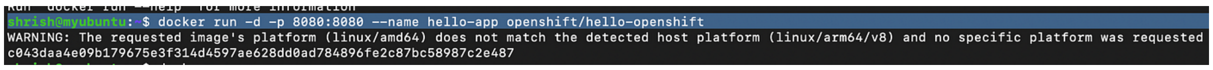
Zoom image will be displayed



```
shrish@myubuntu:~$ docker logs hello-app
exec /hello-openshift: exec format error
```

At first glance, it seemed like a simple issue — but after digging deeper, I realised it was due to an **architecture mismatch**. The image was built for`linux/amd64`, while I was trying to run it on an **ARM-based system** (Mac M2 / ARM VM).

Zoom image will be displayed



```
Run 'docker run --help' for more information
shrish@myubuntu:~$ docker run -d -p 8080:8080 --name hello-app openshift/hello-openshift
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
c043daa4e09b179675e3f314d4597ae628dd0ad784896fe2c87bc58987c2e487
```

Turns out the container was built for x86_64 (amd64), and I was trying to run it on an ARM64 machine.

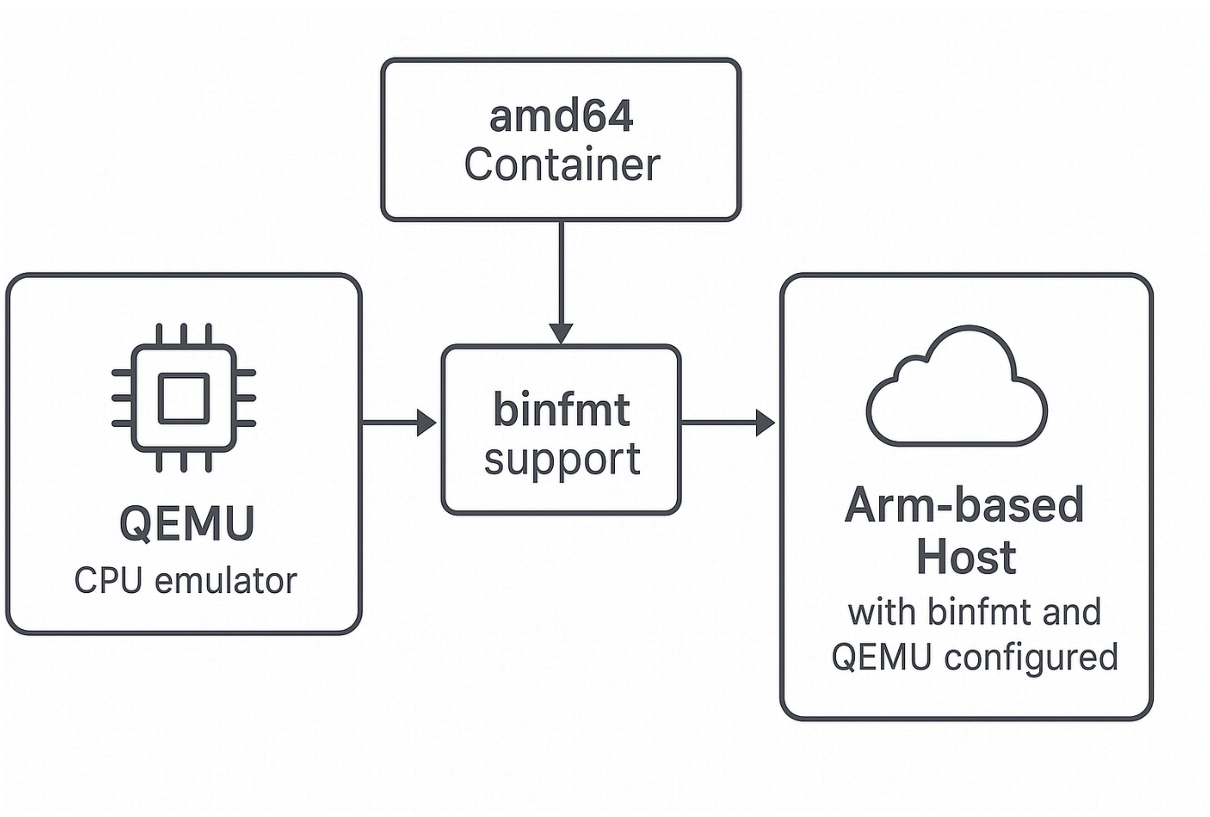
What Was the Issue?

Turns out the container was built for **x86_64 (amd64)**, while my system is running **ARM64**. Basically, the image and the CPU were speaking *two different languages*.

The Fix? QEMU to the Rescue!

That's when I found out about QEMU — an open-source emulator that lets you run binaries built for one architecture on a different one (like `amd64` apps on `arm64` hardware).

Zoom image will be displayed

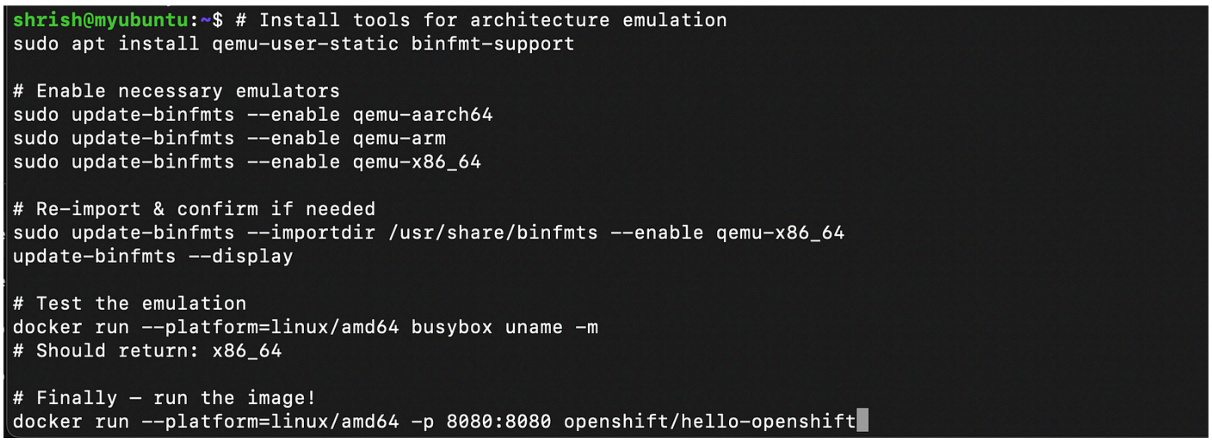


It works with something called `binfmt-support`, which helps the OS understand how to handle foreign binaries (like amd64 on ARM). Super cool, right?

After a bit of research and testing, here's how I solved it:

- Installed `qemu-user-static` and `binfmt-support`
- Enabled `qemu-x86_64` manually using `update-binfmts`
- Pulled and ran the image explicitly using `--platform=linux/amd64`

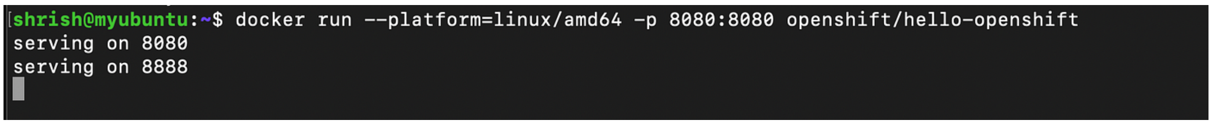
Zoom image will be displayed



```
shrish@myubuntu:~$ # Install tools for architecture emulation
sudo apt install qemu-user-static binfmt-support

# Enable necessary emulators
sudo update-binfmts --enable qemu-aarch64
sudo update-binfmts --enable qemu-arm
sudo update-binfmts --enable qemu-x86_64

# Re-import & confirm if needed
sudo update-binfmts --importdir /usr/share/binfmts --enable qemu-x86_64
update-binfmts --display

# Test the emulation
docker run --platform=linux/amd64 busybox uname -m
# Should return: x86_64

# Finally - run the image!
docker run --platform=linux/amd64 -p 8080:8080 openshift/hello-openshift
```

And yes — **it worked**!

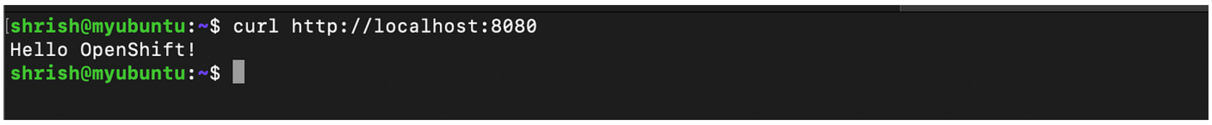
Zoom image will be displayed



```
shrish@myubuntu:~$ docker run --platform=linux/amd64 -p 8080:8080 openshift/hello-openshift
serving on 8080
serving on 8888
```

terminal

Zoom image will be displayed



```
shrish@myubuntu:~$ curl http://localhost:8080
Hello OpenShift!
shrish@myubuntu:~$
```

Opened `http://localhost:8080` and saw Hello OpenShift

What I learned:

- Always check your container’s architecture, especially on ARM machines.
- Tools like QEMU can save the day.
- Sometimes, a little frustration leads to deeper learning.

If you’re working with ARM-based devices, cross-platform containers, or just enjoy debugging deep dives — let’s connect! Would love to hear your stories too.