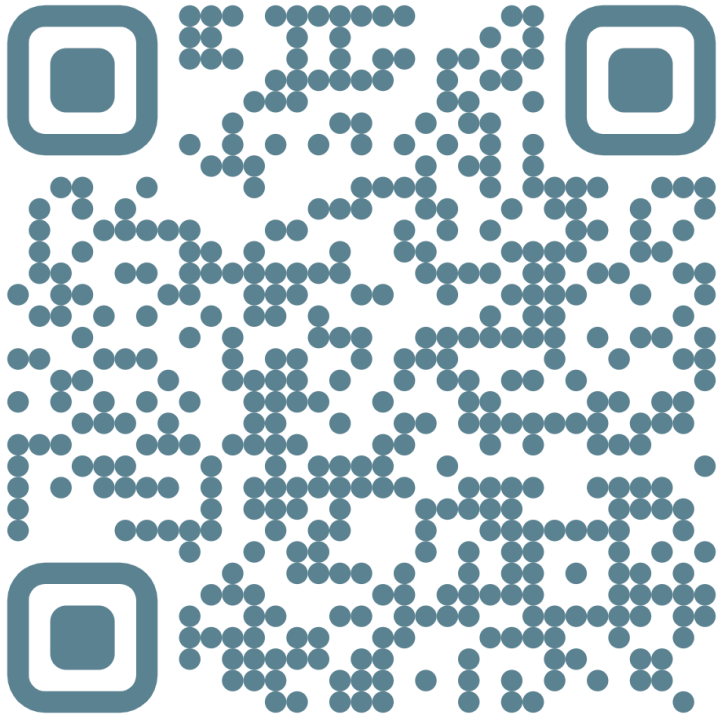
Learn Enough Containers to be Dangerous!

A Containers Workshop

- Ashwin M

About Me

- Ashwin Murali
- Senior DevOps Engineer, ARRC, TII. Abu Dhabi.
- AWS Community Builder - Containers
- 18 years in Tech
- Multiple Series A/B Scale Ups. 15 years on AWS.
- Reach me on Twitter / LinkedIn / Web



Expectations

- Level 100 session
- We have a few small demos
- Some amount of coding involved
- Walk away intelligent! *hopefully*
- Stop me for questions
- Break at roughly halfway mark.

Agenda

- Introduction to Containerization
- Docker 101
- Understanding the Dockerfile
- Docker Volumes & Networks
- Docker Compose
- Containers in Production

Who knows Containers?

Introduction to Containerization

Containers are…

- Lightweight Isolated envs
- Package apps and dependencies
- Provide Consistency and Reliability from one computing env to another
  - It runs on all machines! 😝

Why though?..

- Isolation
- Portability
- Immutability
- Efficiency
- Agility

Isolation…

Isolation of execution via namespaces, PIDs, NICs, File Systems and resource boundaries.

Portability...

Build Once, Run Anywhere - as long as the CPU architecture is the same.

Immutability

Once an image is created, it cannot be changed. Any scale, same image. and the same problems 😖...

Efficiency

Super light weight compared to traditional VMs. More power, less wastage.

Agility

Shorter development and deployment cycles due to easy scaling and resource management.

How did this happen though?

- chroot - isolate folder trees
- namespaces - isolate folder trees, users and processes
- Free BSD Jails - same as chroot, but better!
- cgroups - resource limits

A container…

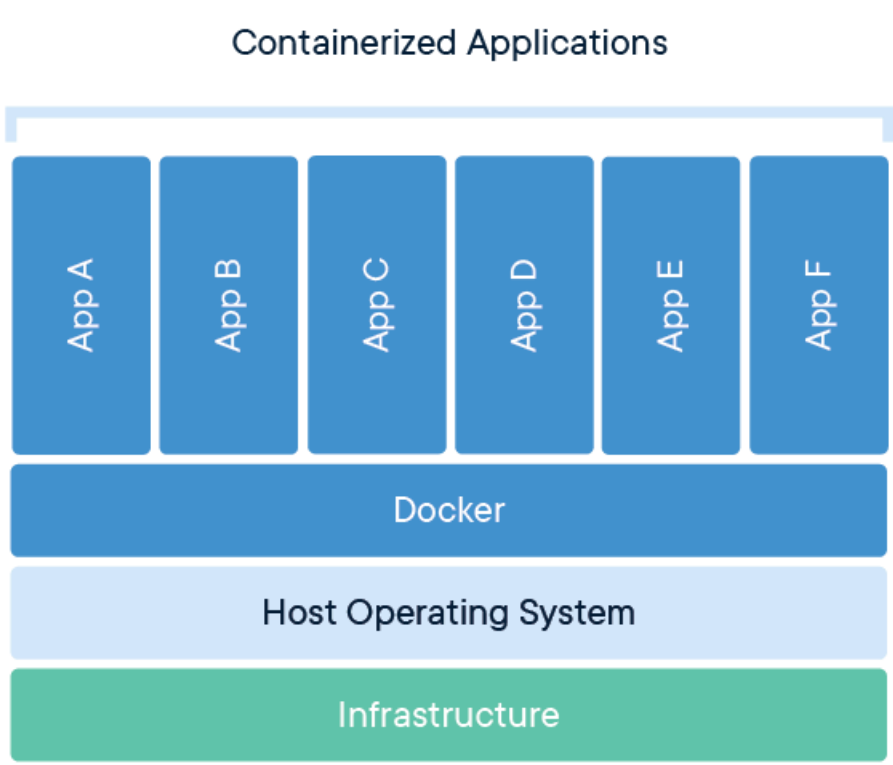
> Can do all the above and a bit more…

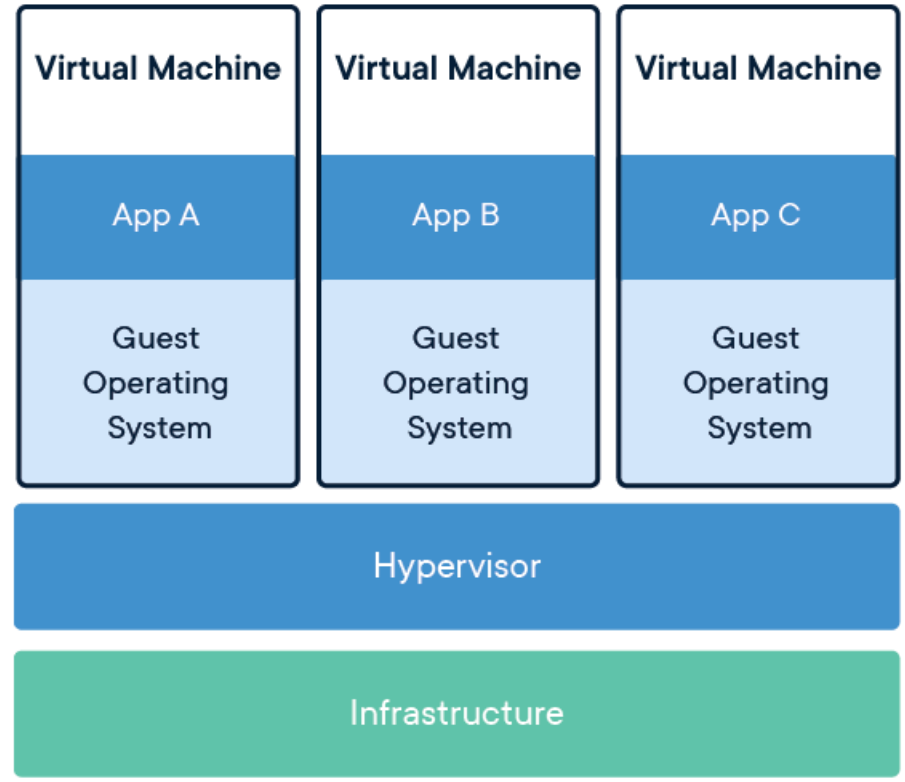
Lets get started?

Docker 101

Docker is an open platform for developing, shipping, and running applications.

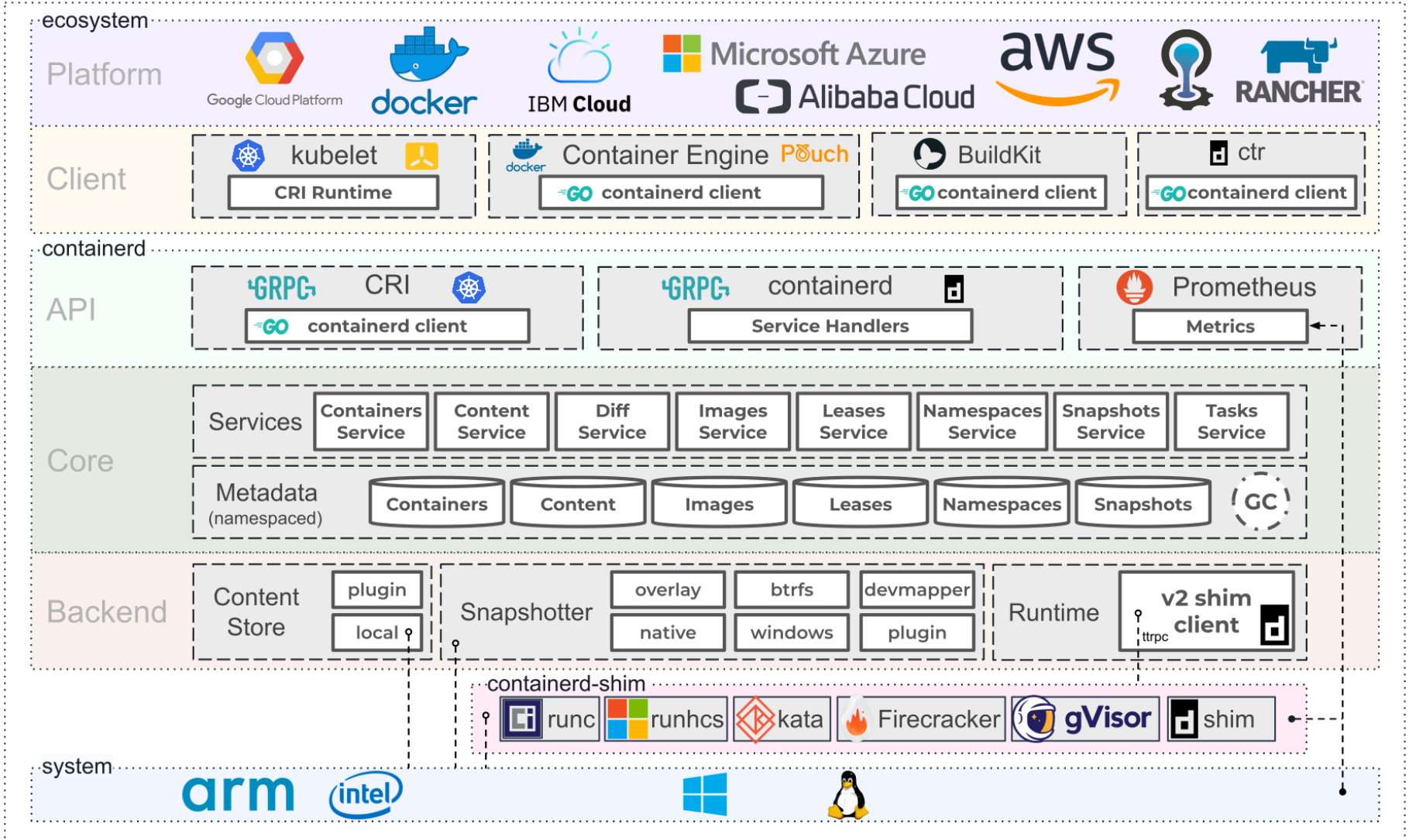
Containers vs VMs





So what is `containerd` ???

- Official Runtime by Docker (Google for OCI Spec)
- Docker == Docker ecosystem (DevTools, Docker Hub, Docker Engine, etc.) + containerd
- Other Alternatives
  - ZeroVM
  - Podman
  - LXD
  - OpenVZ
  - RunC
  - CRI-O


ecosystem
Platform
Google Cloud Platform
docker
IBM Cloud
Microsoft Azure
Alibaba Cloud
aws
RANCHER
Client
kubelet
CRI Runtime
Container Engine
containerd client
BuildKit
containerd client
ctr
containerd client
containerd
API
GRPC
CRI
containerd client
GRPC
containerd
Service Handlers
Prometheus
Metrics
Core
Services
Containers Service
Content Service
Diff Service
Images Service
Leases Service
Namespaces Service
Snapshots Service
Tasks Service
Metadata
(namespaced)
Containers
Content
Images
Leases
Namespaces
Snapshots
GC
Backend
Content Store
plugin
local
Snapshotter
overlay
btrfs
devmapper
native
windows
plugin
Runtime
v2 shim client
ttrpc
containerd-shim
runc
runhcs
kata
Firecracker
gVisor
shim
system
arm
intel

Dockerfile

GET YOUR LAPTOPS OUT!

Dockerfile

```
FROM python:3.9

WORKDIR /app

RUN pip install flask

COPY . .

CMD ["python", "app.py"]
```

app.py

```
#imports
from flask import Flask

app = Flask(__name__)

# ‘/’ URL is bound with hello_world() function.
@app.route('/')
def hello_world():
    return 'Hello World'

# entrypoint
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=4000)
```

Build

```
$ docker build . -t my_python_app
```

Check

```
$ docker images
```

Run

```
$ docker run -p 4000:4000 my_new_app:latest
```

Test

```
$ curl http://127.0.0.1:4000/
Hello World
```

```
$ docker inspect my_new_app:latest
```

```
...
"Cmd": [
    "python",
    "app.py"
],
...
"WorkingDir": "/app",
...
"Layers": [
    "sha256:b10a49b17ae62fcf1c89fbf0473a879599168554d24490433ec580f685c2b879",
    "sha256:973599cf2dadf3755ae7e1322a8fe2b8c0e30bcdee59adee49b71a18c388a1fe",
    "sha256:a974964b27e5246ceec487fc16bd743848f766ea0d62afe6ded2b3ee12ff0699",
    "sha256:d9c6bbb693ea08d5c41175bcf74d9a31971e58f8a79ffb942f31565aead6a08d",
    "sha256:9ce63ba53cb8da4d998a138f4881af9094f2cd20372a77500274a6c63a24a166",
    "sha256:5f895c7ab7df38dfb4af113af3c5d383f55a16317bb2af963c25c5a7cde2e782",
    "sha256:5589e8997c0c0ebb87030f8a90b636c97afc61e6c6f8a13acc0ce6658d984dd5",
    "sha256:3b48824bd4fdafdb56875e3f247491f25335ff61fac12a004d0ee97c9b2f0835",
    "sha256:cf165c849f92e25f85270bb32eff6b0261be25cc57121df57bae47c9cf99ea28",
    "sha256:c39fa1d3d395ede2a82d986d3e04534169a849e2b5c34c57600a9aff96b9bccf",
    "sha256:c691e058b4da09dbbf91d7664dde0265b7afd372e8deddc405081b87f046b1df"
]
...
```

```
$ docker inspect python:3.9
```

```
...
"Layers": [
    "sha256:b10a49b17ae62fcf1c89fbf0473a879599168554d24490433ec580f685c2b879",
    "sha256:973599cf2dadf3755ae7e1322a8fe2b8c0e30bcdee59adee49b71a18c388a1fe",
    "sha256:a974964b27e5246ceec487fc16bd743848f766ea0d62afe6ded2b3ee12ff0699",
    "sha256:d9c6bbb693ea08d5c41175bcf74d9a31971e58f8a79ffb942f31565aead6a08d",
    "sha256:9ce63ba53cb8da4d998a138f4881af9094f2cd20372a77500274a6c63a24a166",
    "sha256:5f895c7ab7df38dfb4af113af3c5d383f55a16317bb2af963c25c5a7cde2e782",
    "sha256:5589e8997c0c0ebb87030f8a90b636c97afc61e6c6f8a13acc0ce6658d984dd5",
    "sha256:3b48824bd4fdafdb56875e3f247491f25335ff61fac12a004d0ee97c9b2f0835"
]
...
```

Sharing images

```
$ docker tag my_new_app:latest <your_docker_hub_username>/my_new_app:latest

$ docker login registry-1.docker.io

$ docker push <your_docker_hub_username>/my_new_app:latest
```

ENV variables

```
ENV APP_PORT=4000 #line 2
...
```

```
...
import os #line 2
...
    app.run(debug=True, host='0.0.0.0', port=os.getenv("APP_PORT", 3000)) #line 13
... # other code
```

Lets rebuild again and inspect the image...

Docker Volumes

- Persistent data storage mechanism for containers
- Data remains even if container is removed
- Three types of volumes:
  - Named volumes: Managed by Docker
  - Bind mounts: Direct mapping to host filesystem
  - tmpfs mounts: Temporary storage in memory (Linux only)

```
# Create and use a named volume
docker volume create my_data
docker run -v my_data:/app/data mysql
```

Where do you use volumes?

- Database storage
- Configuration files
- Shared application data
- Development environments

Docker Networks

- Virtual networks for container isolation
- Built-in DNS resolution
- Multiple network drivers:
  - bridge: Default network driver
  - host: Container uses host’s network
  - overlay: Multi-host networking
  - macvlan: Assign MAC address to container

```
# Create a network
docker network create my_network

# Run container in network
docker run --network my_network mysql

# Connect existing container
docker network connect my_network container_name
```

transition: fade-out layout: end

Thank you!