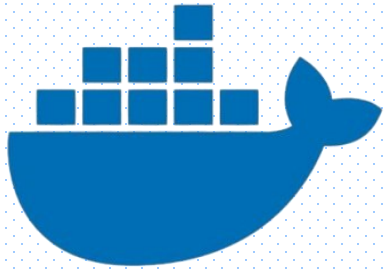
DSRL
Distributed Systems
Research Laboratory

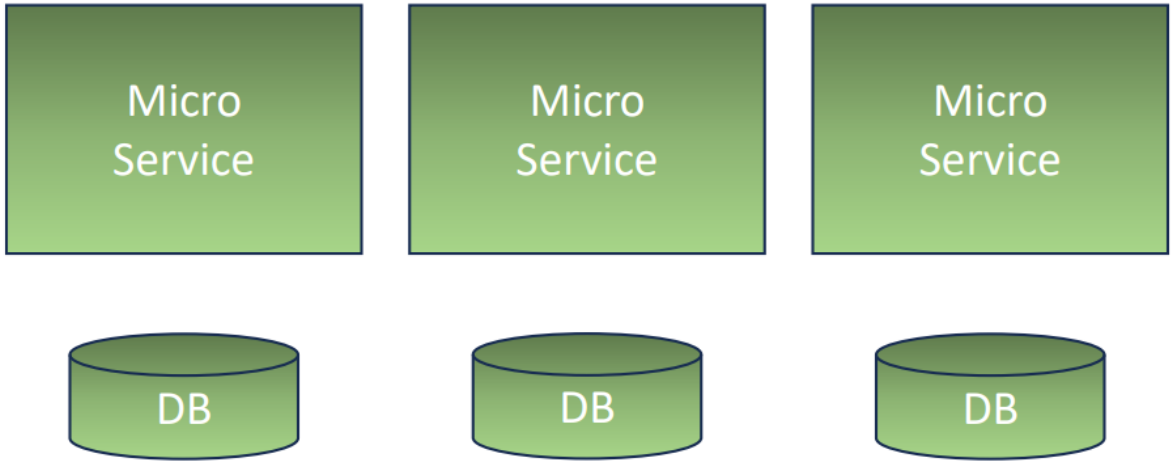


Deployment with docker

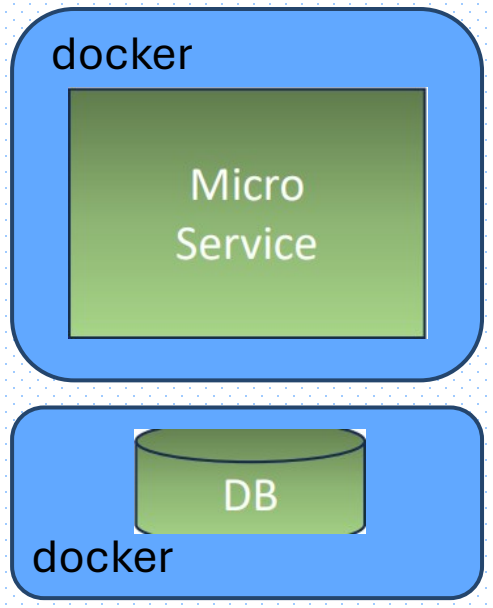
Technical University of Cluj-Napoca

TECHNICAL UNIVERSITY OF CLUJ-NAPOCA ROMANIA

Why Docker?



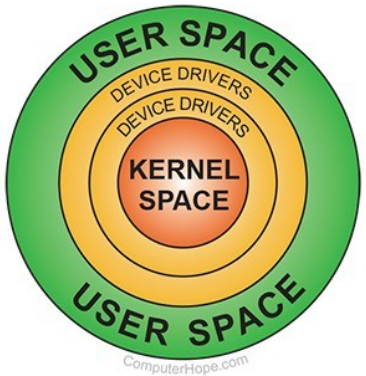
- Modern applications -> **microservices**
  - Made of multiple small services
- Each microservice runs in its own environment
  - Dependencies can conflict
- Docker solves this by packaging apps and their dependencies into **images**, which run as **containers**



From Virtual Machines to Containers

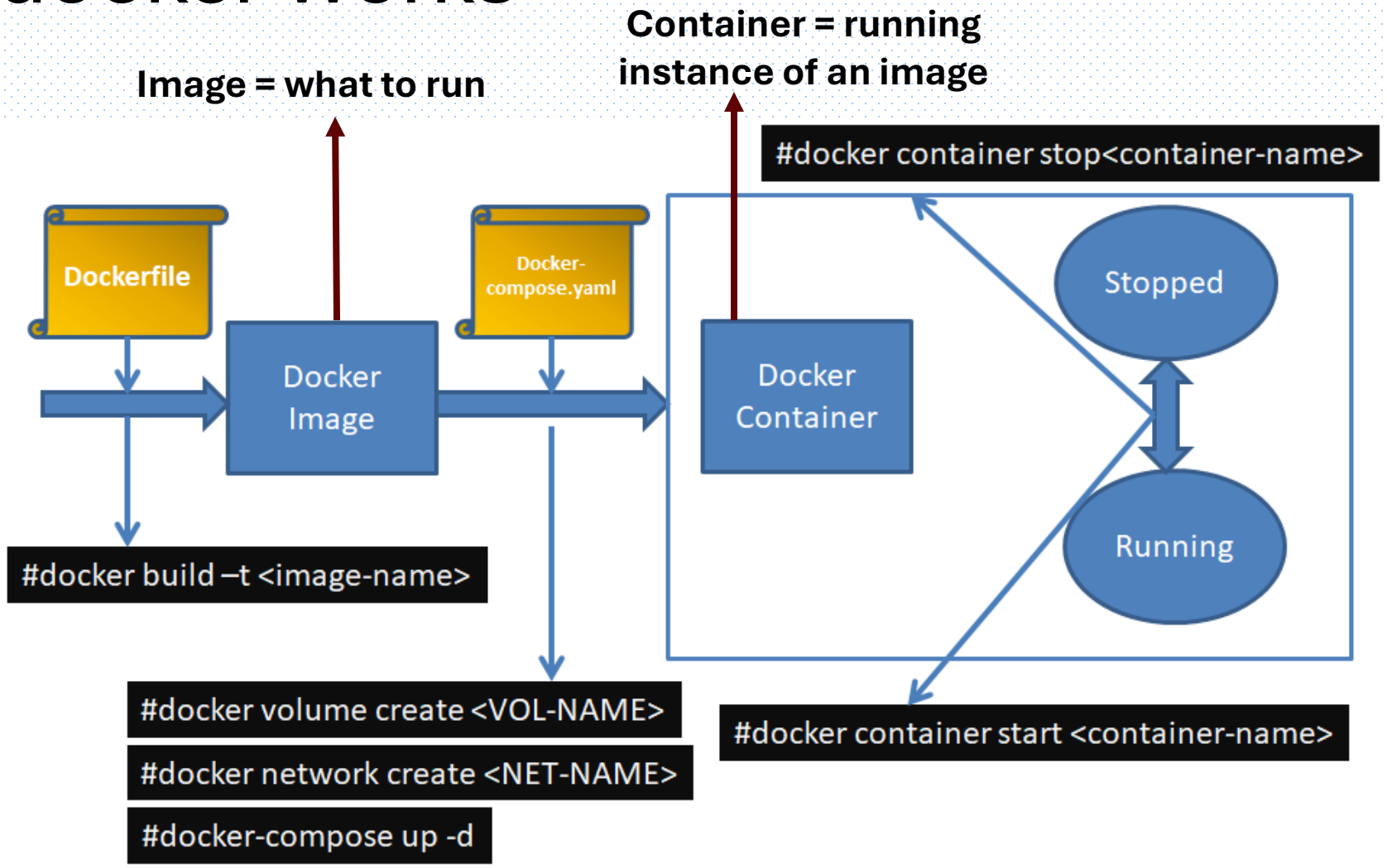
- **Virtual Machine** – emulation of a computer system
  - Full Operating System + apps -> heavy, slow to start
  - Role: full OS emulation
- **Container** – running as a process in the user space
  - Shares Operating System with its host -> light, fast, efficient
  - Role: process isolation
- Multiple containers can run on one machine

From Virtual Machines to Containers



Features	VM	Docker
Host OS	Any	Linux – based (if installed on Windows it installs a VM with Linux)
Guest OS	Any	Linux – based (it uses the kernel of the host operating system)
Sandboxing	Full isolation	Can access host through shared filesystem (such as docker-volume)
HW Resource requirements	High (like the physical machine emulated)	Low (many containers can run on the same physical machine)

How docker works



Docker image and Dockerfile

- **Dockerfile –** used to create an image
- Builds a repeatable environment for your app

Build an image from a Dockerfile

```
#docker build –t <image-name>
```

See all images created

```
#docker image ls
```

```
FROM node:18
WORKDIR /app
COPY package.json ./
RUN npm install
COPY . .
CMD npm start
EXPOSE 3000
```

Dockerfile

Docker Container Lifecycle

Run a container from an image

```
#docker run –d –p 8080:8080 myapp
```

List all containers

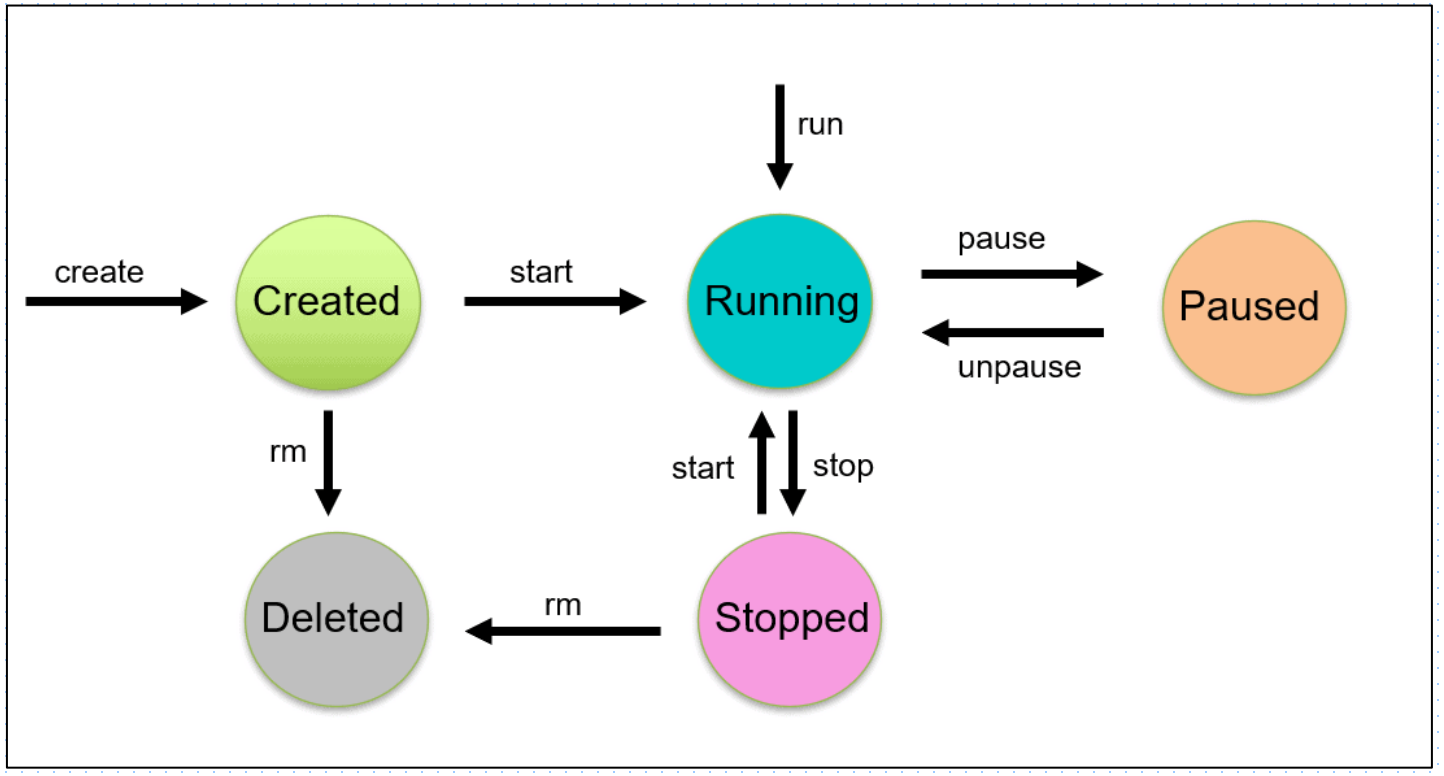
```
#docker ps
```

Stop a container

```
#docker container stop myapp
```

Start an existing container

```
#docker container start myapp
```



Docker Volumes and Networks

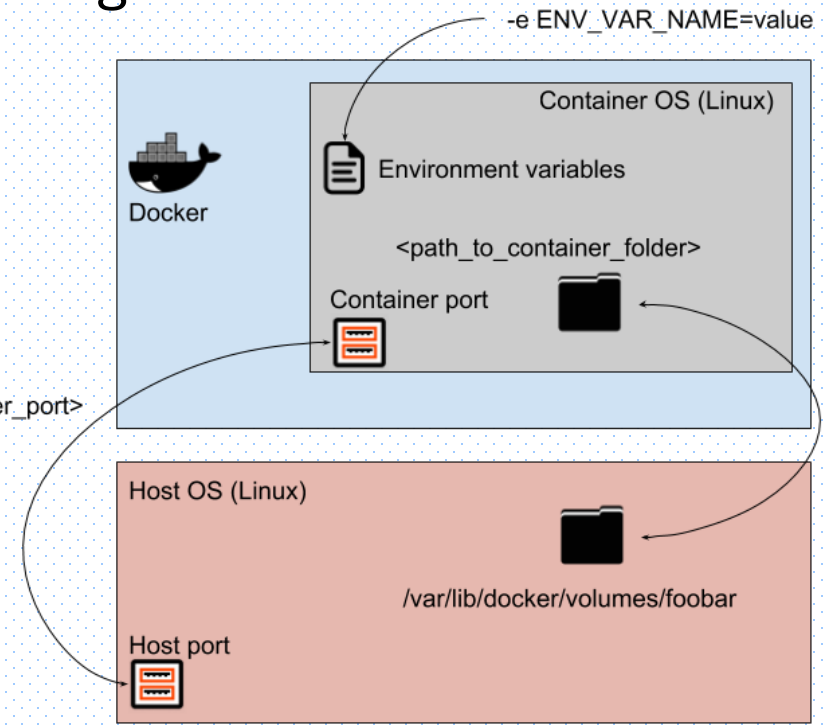
- **Volume =** shared folder between container and host
- **Network =** containers can connect together through it

Create a volume

```
#docker volume create mydata
```

Create a network

```
#docker network create mynet
```



Docker Containers

- **docker-compose.yml –** define multiple containers
- Specify volumes, ports, image, all from one file

Run all services from the docker compose

```
#docker-compose up -d
```

```
services:
  backend:
    build: <image-name>
    ports: "8080:8080"
  frontend:
    build: <image-name>
    ports: "3000:3000"
```

docker-compose.yml