Hands-on Exercise 1.1: VM vs Container Comparison

Objective

Compare performance and resource usage between virtual machines and containers using Ubuntu as the base operating system.

Prerequisites

- Computer with Docker installed
- Virtual machine software (VirtualBox/VMware/Hyper-V) with Ubuntu VM
- Basic terminal/command-line knowledge

Exercise Instructions

1. Start a Virtual Machine with Ubuntu

```
# Note time before starting VM
date +%s

# Start Ubuntu VM from instructor's machine
# Instructor will demonstrate this step

# Note time after VM is ready for use
date +%s

# Calculate startup time
# End time - Start time = VM startup time in seconds
```

2. Start a Docker Container with Ubuntu

```
# Note time before starting container
date +%s

# Pull Ubuntu image (if not already downloaded)
docker pull ubuntu:22.04

# Start Ubuntu container with interactive shell
docker run -it --name ubuntu-container ubuntu:22.04 bash

# Note time when container prompt appears
date +%s

# Calculate startup time
# End time - Start time = Container startup time in seconds
```

3. Compare Resource Usage (RAM, CPU)

For the VM

```
# On host system, check VM resource usage
# Instructor will show VM resource monitor

# From inside VM, check memory usage
free -m

# Check CPU usage
top -n 1
```

For the Container

```
# From another terminal, check container resource usage
docker stats ubuntu-container

# From inside container, check memory usage
free -m

# Check container CPU usage
top -n 1
```

4. Compare Startup Times

```
# Record and compare the startup times from steps 1 and 2
echo "VM startup time: XX seconds"
echo "Container startup time: XX seconds"
```

5. Run Identical Workloads and Measure Performance

Prepare Test Workload

```
# Create test script for both environments
cat > test-workload.sh << 'EOF'
#!/bin/bash
echo "Starting CPU test..."
start_time=$(date +%s.%N)
for i in {1..100}; do
  echo "Iteration $i"
  for j in {1..10}; do
    echo $j | md5sum > /dev/null
  done
done
end_time=$(date +%s.%N)
execution_time=$(perl -e "print $end_time - $start_time")
echo "CPU test completed in $execution_time seconds"

echo "Starting I/O test..."
start_time=$(date +%s.%N)
for i in {1..5}; do
  dd if=/dev/zero of=test.file bs=1M count=100 oflag=direct
  sync
  rm test.file
done
end_time=$(date +%s.%N)
execution_time=$(perl -e "print $end_time - $start_time")
echo "I/O test completed in $execution_time seconds"
EOF

# Make script executable
chmod +x test-workload.sh
```

Run Test on VM

```
# Copy script to VM
# Instructor will demonstrate this

# Execute script in VM and record time
./test-workload.sh
```

Run Test on Container

```
# Copy script to container
docker cp test-workload.sh ubuntu-container:/

# Execute script in container and record time
docker exec -it ubuntu-container bash -c "chmod +x /test-workload.sh && /test-workload.sh"
```

6. Cleanup

```
# Stop and remove container
docker stop ubuntu-container
docker rm ubuntu-container

# Instructor will shut down VM
```

Expected Results and Analysis

Metric	Virtual Machine	Container	Comparison
Startup Time			Containers typically start in milliseconds vs. minutes for VMs
Memory Usage			Containers share host kernel, requiring less memory
CPU Usage (idle)			Containers have less overhead
Workload Performance			Containers often have near-native performance
Disk Space			Container images are much smaller than VM images

Key Observations

1. Containers start significantly faster than VMs
2. Containers use fewer resources (memory, CPU, disk)

3. Containers provide comparable or better performance for many workloads
4. VMs provide stronger isolation but with higher resource costs
5. Containers share the host kernel, while VMs run their own kernels

Discussion Questions

1. When would you prefer to use containers over VMs?
2. When might VMs be more appropriate than containers?
3. How do these differences impact development and deployment workflows?
4. What are the security implications of these architectural differences?