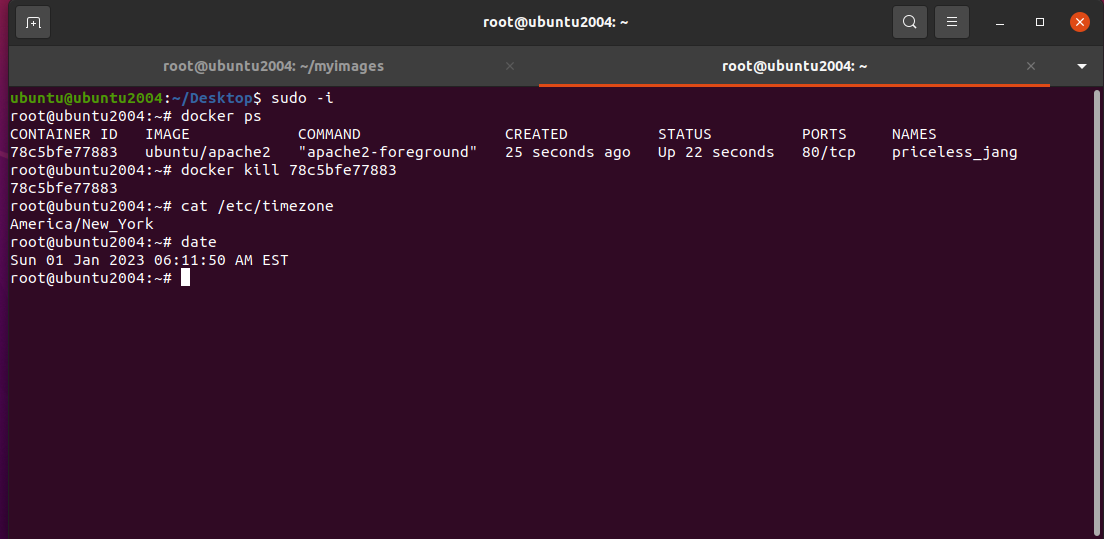
*PART 1 When working with containers (typically Docker), it's very common for base images to be set with some default timezone. However, in an environment where containers are deployed around the globe, it is sometimes required to have containers that are set to the timezone of their host machines.*

*Your assignment is to find the most robust, universal and generic way to do the following:*

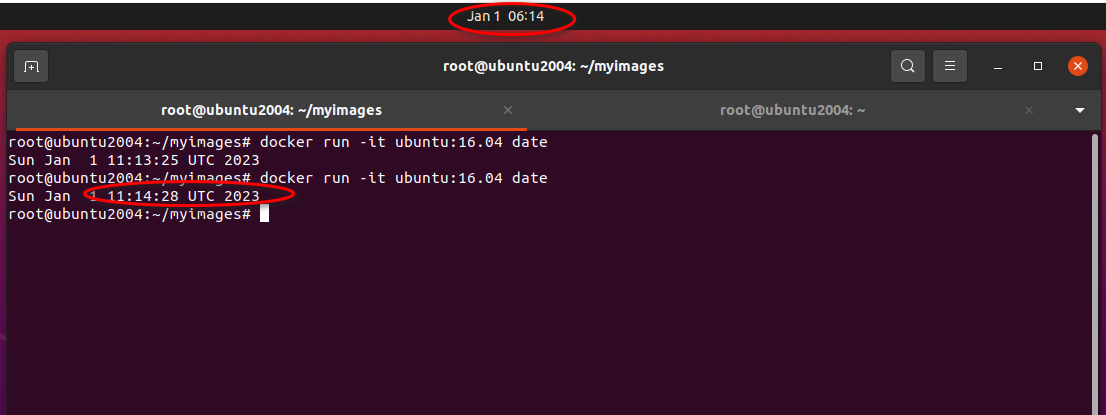
*1. Given a microservice-based application running in Docker containers, set the service containers to the same timezone as the host that runs them. The services use a few different base images:*

1. *ubuntu:16.04*
2. *ubuntu:18.04*
3. *alpine:latest*
4. *fedora:latest*
5. *centos:8*

**Answer:**

Example - I set up a VM ( on VBox ) – it has a local timezone of new York:  


If I run the docker command for ubuntu 16.04 with no adjustments – I get the following:



Notice the difference in times – due to the timezones. Base host is in New York, But the docker container is running in UTC.

If however, one runs as follows:  
docker run -v /etc/timezone:/etc/timezone -v /etc/localtime:/etc/localtime <base image>

Note – if you use docker-compose.yml instead – this would be needed:

services:

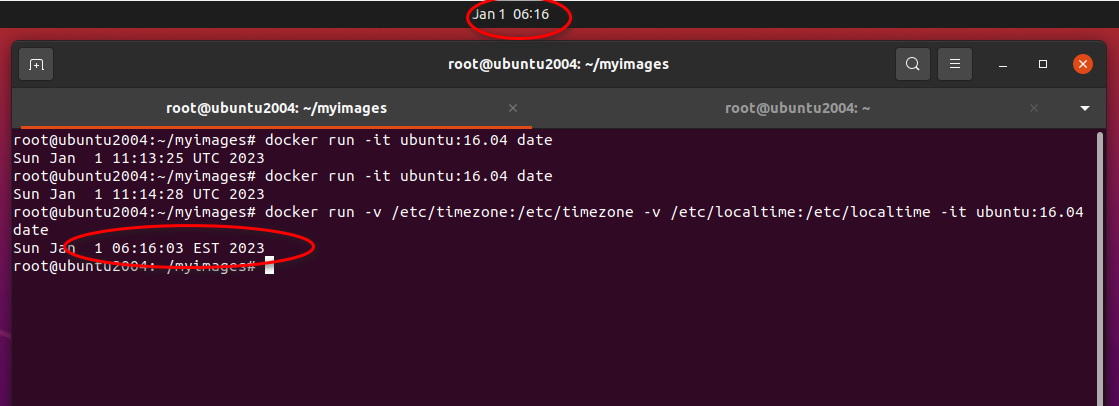
service\_name:

volumes:

- "/etc/timezone:/etc/timezone:ro"

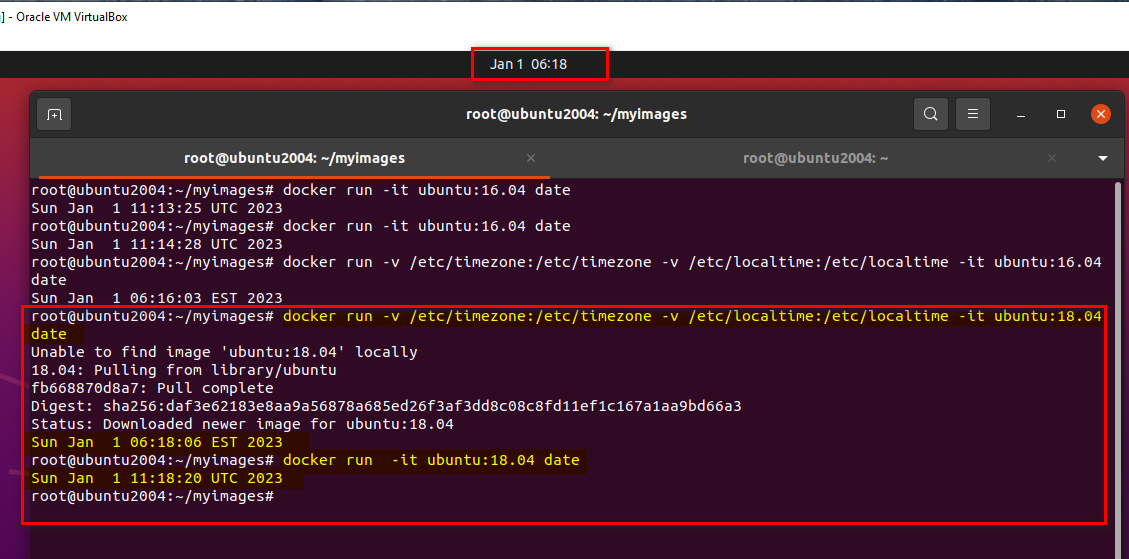
- "/etc/localtime:/etc/localtime:ro"

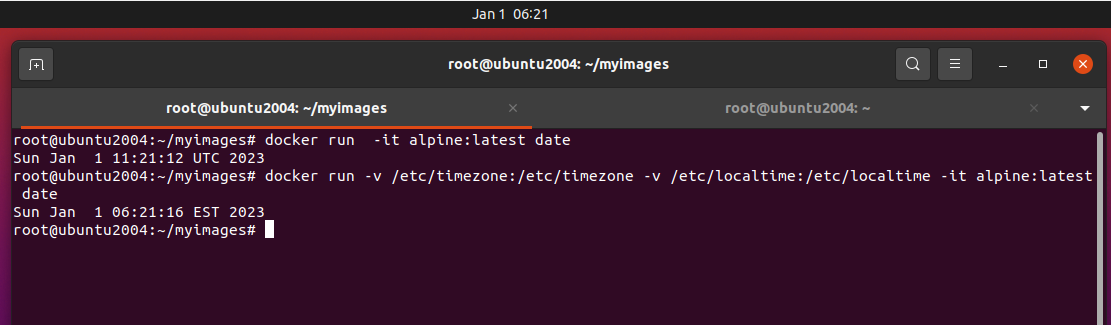
Then the timezone of the docker container will be the same as the host:



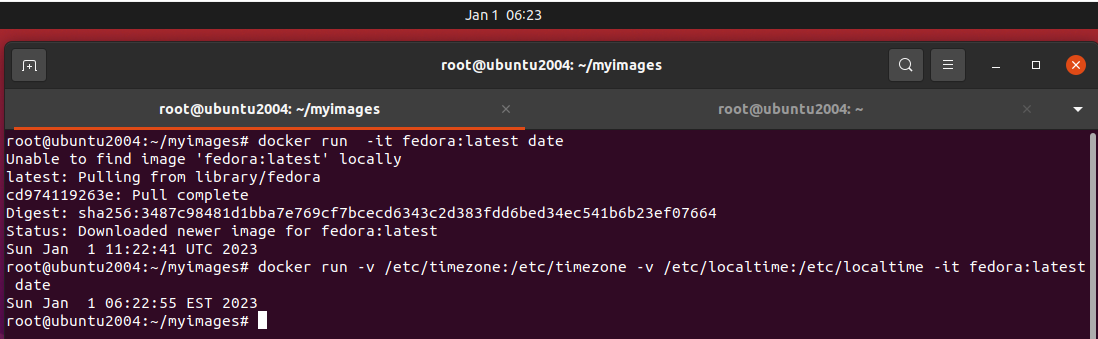
We can now see that the date and time and timezone has been adjusted – to that of the host machine.

The same trick works with 18.04:

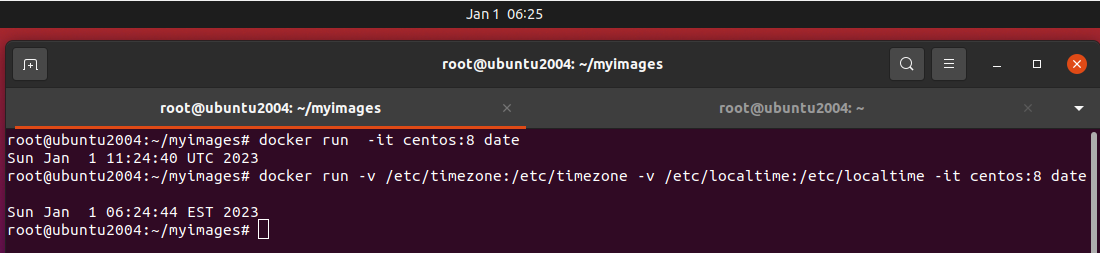


Same also works for Alpine:  


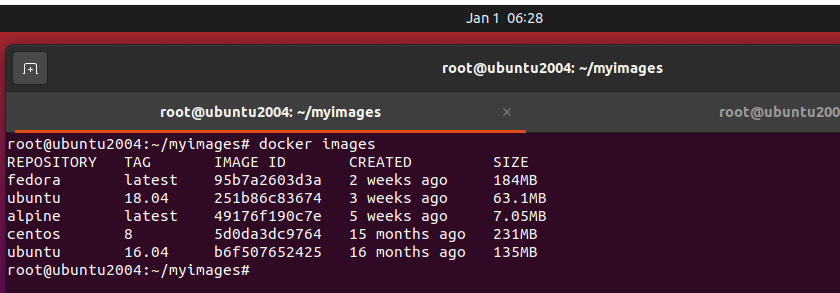
As is also true of fedora:



And also centos:8



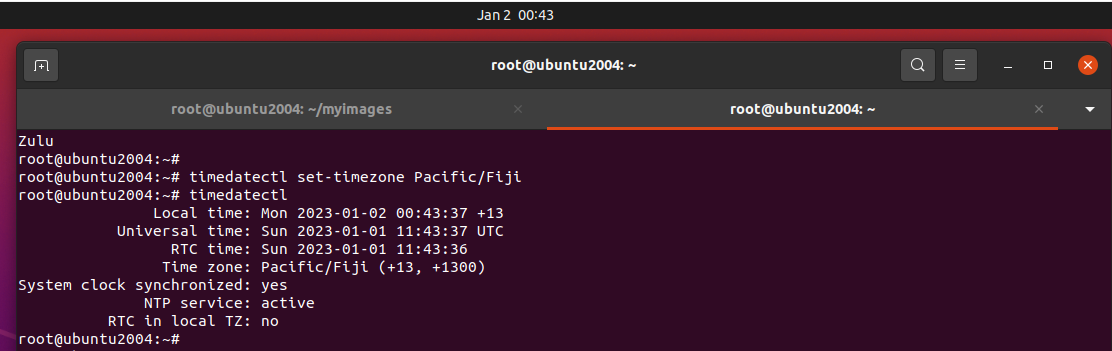
Proof of images downloaded:



*2. For specific services that are built with an environment variable OVERRIDE\_TZ\_EST=1, do not use the timezone of the host, but rather always Eastern Time. Note that these services may use any of the base images described above.*

Answer:

To show how this works, firstly, I set my Host machine to some obscure timezone – in our case Fiji (!)



In order to run now the docker containers – with the EST timezone – fixed – I will try and run the following:

﻿root@ubuntu2004:~# date

Mon 02 Jan 2023 08:44:06 AM +13

root@ubuntu2004:~# for i in $(echo "centos:8 fedora alpine ubuntu:16.04 ubuntu:18.04"); do echo -n "$i :" ;docker run -v /etc/timezone:/etc/timezone -v /etc/localtime:/etc/localtime -e TZ=EST $i date; done

centos:8 :Sun Jan 1 14:44:10 EST 2023

fedora :Sun Jan 1 14:44:11 EST 2023

alpine :Sun Jan 1 19:44:12 UTC 2023

ubuntu:16.04 :Sun Jan 1 19:44:13 EST 2023

ubuntu:18.04 :Sun Jan 1 19:44:15 EST 2023

Note – the centos:8 and fedora – produce the correct time and timezone, Alpine – doesn’t have any effect, and the two ubuntu’s set the correct timezone – but not the correct date/time

For Alpine – we can create a new image – using the following Dockerfile:  
  
FROM alpine:latest  
RUN apk add --no-cache tzdata  
ENV TZ=EST

And when we run it now – we will get the correct response:

root@ubuntu2004:~/myimages# docker run --rm myalpine date

Sun Jan 1 14:51:31 EST 2023

A similar solution, can be used for the ubuntu containers:

﻿

FROM ubuntu:16.04

ENV TZ ‘EST’

RUN echo $TZ > /etc/timezone && \

apt-get update && apt-get install -y tzdata && \

rm /etc/localtime && \

ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && \

dpkg-reconfigure -f noninteractive tzdata && \

apt-get clean

and

FROM ubuntu:18.04

ENV TZ ‘EST’

RUN echo $TZ > /etc/timezone && \

apt-get update && apt-get install -y tzdata && \

rm /etc/localtime && \

ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && \

dpkg-reconfigure -f noninteractive tzdata && \

apt-get clean

If we now run the previous command with the updated images:

root@ubuntu2004:~/myimages# date

Mon 02 Jan 2023 09:10:07 AM +13

root@ubuntu2004:~/myimages# for i in $(echo "centos:8 fedora myalpine myubuntu16.04 myubuntu18.04"); do echo -n "$i :" ;docker run -v /etc/timezone:/etc/timezone -v /etc/localtime:/etc/localtime -e TZ=EST $i date; done

centos:8 :Sun Jan 1 15:10:10 EST 2023

fedora :Sun Jan 1 15:10:11 EST 2023

myalpine :Sun Jan 1 15:10:13 EST 2023

myubuntu16.04 :Sun Jan 1 15:10:14 EST 2023

myubuntu18.04 :Sun Jan 1 15:10:15 EST 2023

**Part 2**  
*Please consider the following scenario.   
Some of our developers, who for some reason are not running Linux on their laptops, test and debug the app in a virtual machine (specifically using Docker Desktop). The app running in some of these virtual machines experience a noticeable clock drift -- the time reported by the services inside the containers is getting inexact over time, even though the laptop clocks are correct.   
Unsure what to do, the R&D manager asks for your advice.   
How would you advise to address the problem?*

Answer:

My gut instinct is to somehow install within the VMs an NTP client linked to either the host (i.e. put an an NTP server on the host) – or to an internet NTP server.

There is a lot of discussion regarding this on the internet how clock drift can occur within VMs or within containers running within – here are some of the summaries:

From: <https://blog.jverkamp.com/2017/11/15/clock-drift-in-docker-containers/>   
The person found that their docker container was running with a clock drift.   
Eventually, the person found that running the container in a privileged mode helped solve the issue:  
docker run --rm --privileged alpine hwclock -s

From: <https://www.thegrussalo.com/2021/01/docker-container-time-drift-using-wsl2.html>   
The person also found clock drift. The given solution to the person was to reboot – but this was unacceptable to them – as they had recently rebooted the pc. The person was running a linux subsystem under windows – and found that a github issue (<https://github.com/microsoft/WSL/issues/4149> ) was reported about this. To get around this – the solution was to run the command “wsl --shutdown” from a command window – when this happened, the Docker desktop requested that it isn’t working, and requires a restart.

From: <https://www.docker.com/blog/addressing-time-drift-in-docker-desktop-for-mac/> - A blog from docker itself. The blog goes into great detail on why the issue happens – and even suggests using an NTP client/server combination ( as I suggested earlier ). In addition, the blog suggests altering the kernel ‘tick’, or – running “settimeofday” at frequent intervals.

Finally, looking at stackoverflow - <https://stackoverflow.com/questions/24551592/how-to-make-sure-dockers-time-syncs-with-that-of-the-host> - also talks about the clock drift issues. The post includes comments from people that found that their times where getting out of sync when someone put their PCs into sleep mode, and then woke them up. The VMs running on the laptops, didn’t adjust their times after waking up – and this caused drift for some people. The post also talks about using the /etc/timezone volume mount – to sync between the containers and the host ( like I answered in the first part ). The post goes on to report multiple solutions – that we have already mentioned above.

Based on all of the above I would recommend – firstly to try a privileged mode on one of the laptops – and see over time – if the time drift continues.

Failing that, I would add to the docker containers – an ntp client – syncing to either the host or an internet ntp server – and again monitor the drift over time.

If that fails, I would continue looking – for

1. possible bugs in the VM / host implementation ( online / github ), and
2. Perhaps getting all developers working with the same dev framework. ( costly, but might be worth it in the long run ).

Mark Prager 2/1/2023.