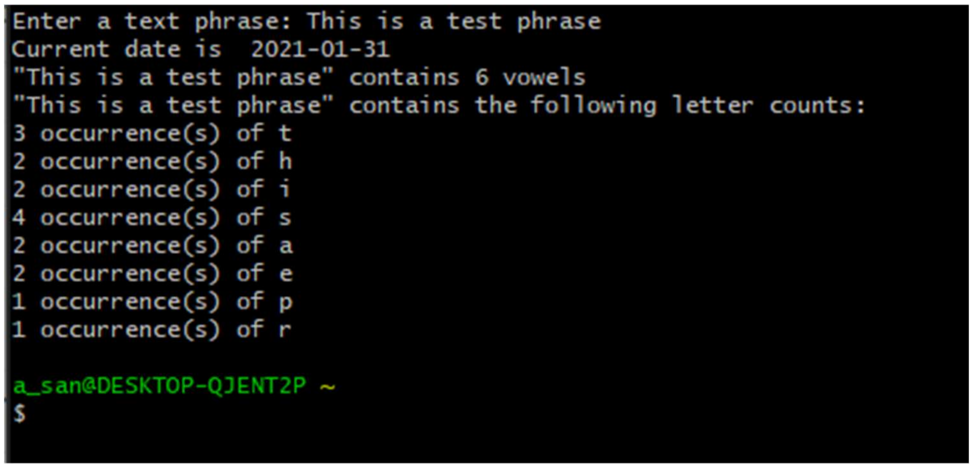
Docker Lab

This lab will provide an opportunity for “hands on” practice of concepts learned from review of the videos and supporting material on the topic in focus. The lab is considered “open book” – you may refer to the relevant videos and reference material (as well as use internet searches) as part of unit completion. The labs are self-paced and will be completed on your own time. While the labs will not be graded, they will provide participants with a chance to reinforce ongoing learning and will help to better position an individual for overall program success. A lab solution will be provided for review – however, it is recommended that you only utilize to check your work after completing the lab exercises.

Build a Python-based image and container for hosting/executing a Python app:

- In your browser, navigate to the public GitHub repository at https://github.com/KernelGamut32/dockerlab-repo1
- Fork the repository to create a copy under your GitHub account
- Using the URL for your forked version of the repository, clone the repository locally
- This is a simple Python app containing string functions and a date formatting operation



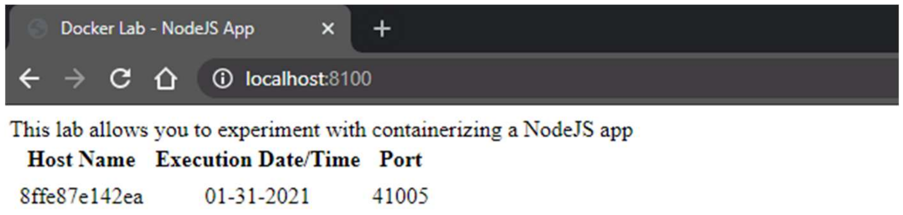
```
Enter a text phrase: This is a test phrase
Current date is  2021-01-31
"This is a test phrase" contains 6 vowels
"This is a test phrase" contains the following letter counts:
3 occurrence(s) of t
2 occurrence(s) of h
2 occurrence(s) of i
4 occurrence(s) of s
2 occurrence(s) of a
2 occurrence(s) of e
1 occurrence(s) of p
1 occurrence(s) of r

a_san@DESKTOP-QJENT2P ~
$
```

- Using Docker Hub (https://hub.docker.com), search for a Python image and use the provided information on usage to create a Dockerfile for building a new docker image
- Use docker command(s) to build a new image using a tag of your choosing
- Use docker command(s) to list details for all local images, including the newly created image
- Using the local image, create a new container for hosting the Python runtime to support execution of the Python app
- The app prompts the user for input – make sure the created container allows for interaction from the user for responding to prompt(s)
- NOTE: You will have to use either Cygwin, the standard command-line, or Git CMD; Git Bash will not work
- Use docker command(s) to list details for all containers, including the newly created container
- Use docker command(s) to execute the python app in the container a second time without creating a new container (i.e., use the existing container for repeat execution)
- Use docker command(s) to list the folder contents for the container

Build a NodeJS-based image and container for hosting/executing a NodeJS app:

- In your browser, navigate to the public GitHub repository at https://github.com/KernelGamut32/dockerlab-repo2
- Fork the repository to create a copy under your GitHub account
- Using the URL for your forked version of the repository, clone the repository locally
- This is a simple NodeJS app containing tabular display (like the app demonstrated on Pluralsight)



This lab allows you to experiment with containerizing a NodeJS app

Host Name	Execution Date/Time	Port
8ffe87e142ea	01-31-2021	41005

- Using Docker Hub (https://hub.docker.com), search for a Node image and use the provided information on usage to create a Dockerfile for building a new docker image; you may also find https://nodejs.org/en/docs/guides/nodejs-docker-webapp/ helpful
- Use docker command(s) to build a new image using a tag of your choosing
- Use docker command(s) to list details for all local images, including the newly created image
- Using the local image, create a new container for hosting the NodeJS runtime to support execution of the NodeJS app
- The app is configured to run at port 41005 (i.e., using http://localhost:41005 in the browser) – for the new container, create a port mapping of 8100 to 41005 (i.e., user will be able to access the app in the browser using http://localhost:8100 when the container is running)
- Use docker command(s) to check logs for the newly created container
- Use docker command(s) to list details for all containers, including the newly created container
- Ensure that the container continues running – open different browser tabs/sessions and hit the same URL (http://localhost:8100) to confirm ongoing application operation
- Use docker command(s) to view the contents of the deployed server.js file in the container

Cleanup:

- Use docker command(s) to delete both containers
- Use a single docker command to delete all local images