Lesson 4 Demo 19: Create a Multi-Container Pod which Access the Same Volume

This section will guide you to:

* Create a multi-container pod in which the containers access the same volume

This lab has one sub-section, namely:

1. Creating a multi-container pod in which the containers access the same volume

**Note:** If you don’t have an existing Kubernetes cluster, refer to the Demo 1.1 of Lesson 1.

**Step 1:** Creating a multi-container pod in which the containers access the same volume

* Start the kubernetes cluster in the lab
* We will now create a pod that runs two containers. The two containers share a volume that they can use to communicate. The configuration file for the pod is as shown below:

*cat> two-container-pod.yaml*

*apiVersion: v1*

*kind: Pod*

*metadata:*

*name: two-containers*

*spec:*

*restartPolicy: Never*

*volumes:*

*- name: shared-data*

*emptyDir: {}*

*containers:*

*- name: nginx-container*

*image: nginx*

*volumeMounts:*

*- name: shared-data*

*mountPath: /usr/share/nginx/html*

*- name: debian-container*

*image: debian*

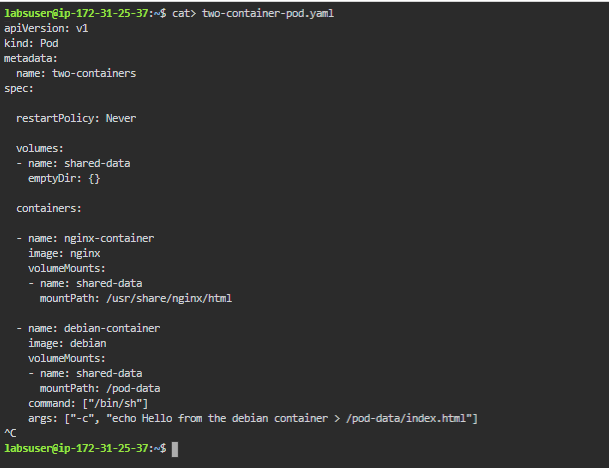
*volumeMounts:*

*- name: shared-data*

*mountPath: /pod-data*

*command: ["/bin/sh"]*

*args: ["-c", "echo Hello from the debian container > /pod-data/index.html"]*

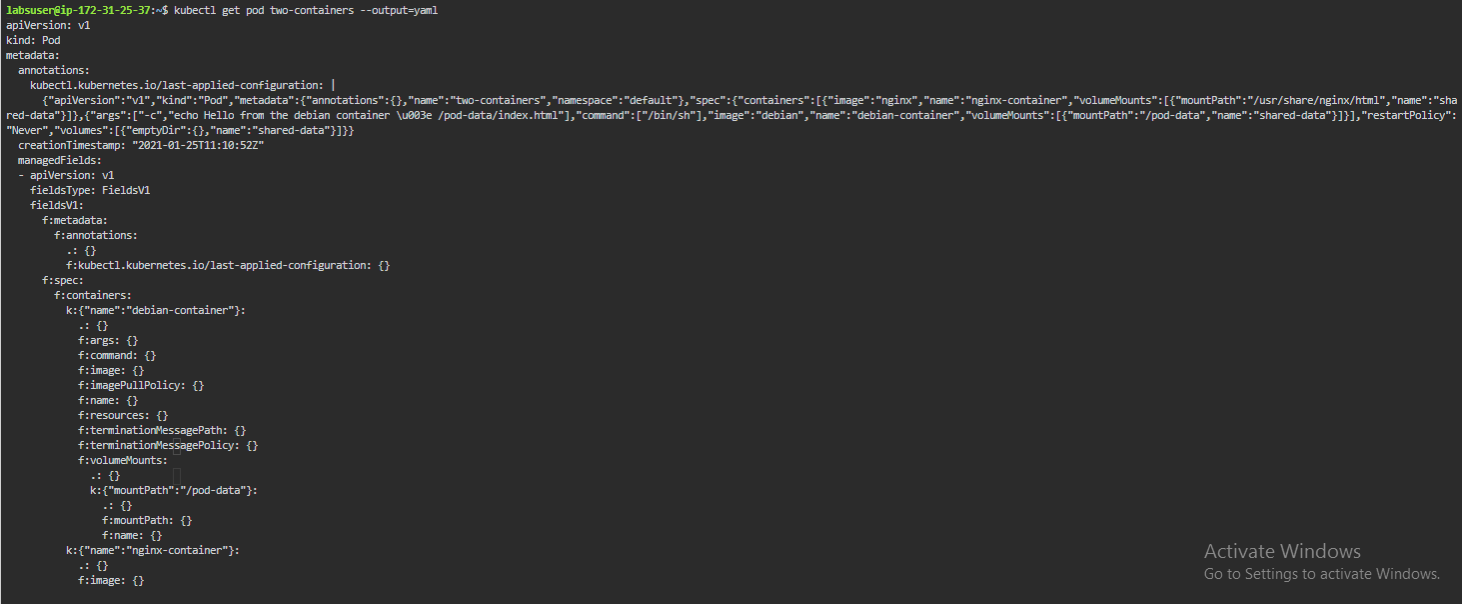
  
In the configuration file, you can see that the pod has a volume named shared-data

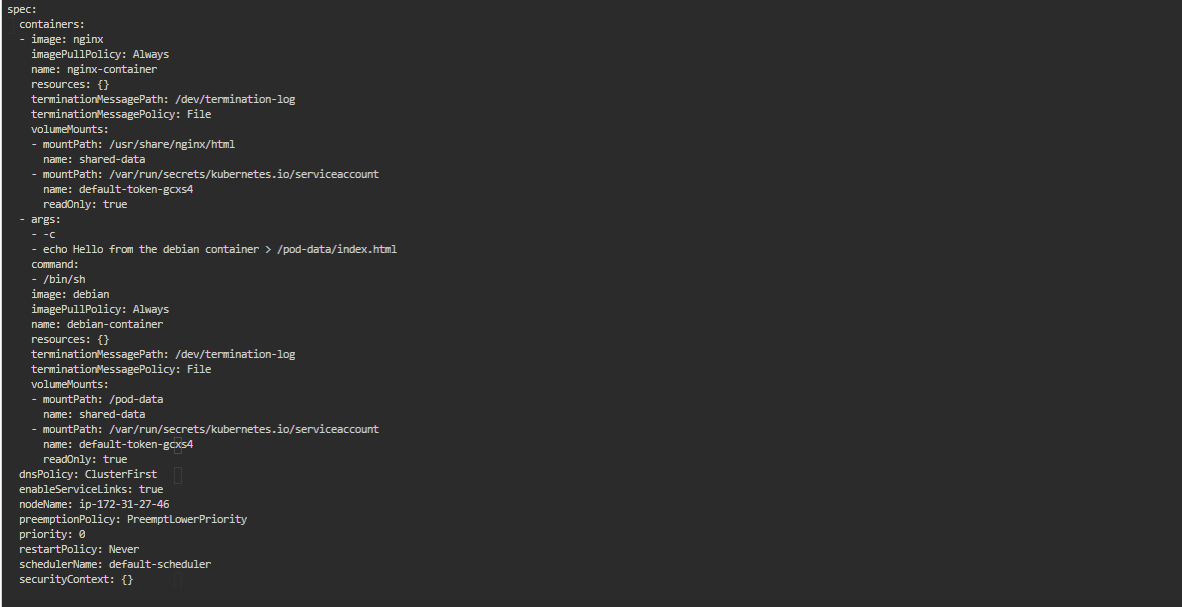
* Now create the pod and the two containers as shown below:

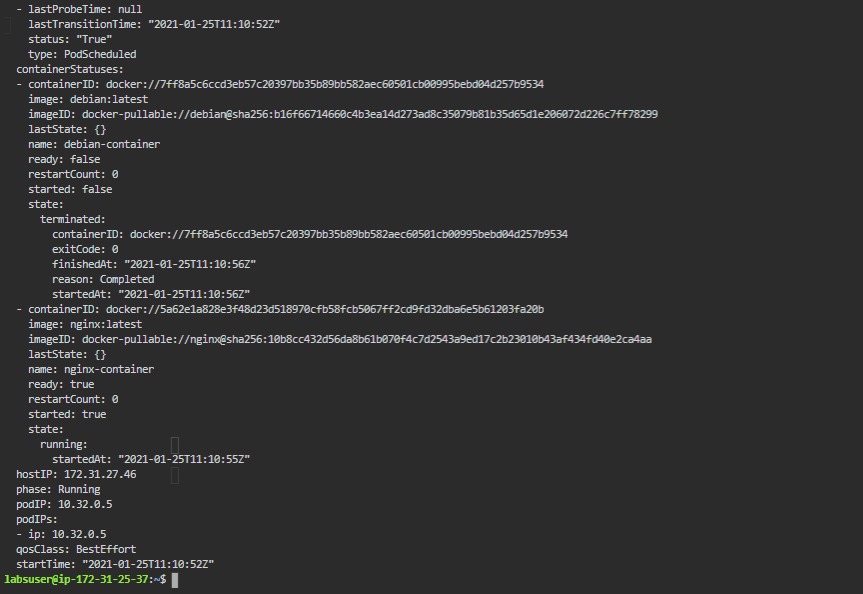
*kubectl apply -f two-container-pod.yaml  
*

* You can verify the information about the pod and the containers using the command below:

*kubectl get pod two-containers --output=yaml*







You can see that the debian container has been terminated and the nginx container is still running

This is how two or more containers can access and share a single volume