Occasionally, it's necessary to customize how containers interact with the underlying security mechanisms present on the operating systems of Kubernetes nodes. The `securityContext` attribute in a pod specification allows for making these customizations. In this lesson, we will briefly discuss what the securityContext is, and demonstrate how to use it to implement some common functionality.

Relevant Documentation

- https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Lesson Reference

First, create some users, groups, and files on both worker nodes which we can use for testing.

```
sudo useradd -u 2000 container-user-0
sudo groupadd -g 3000 container-group-0
sudo useradd -u 2001 container-user-1
sudo groupadd -g 3001 container-group-1
sudo mkdir -p /etc/message/
echo "Hello, World!" | sudo tee -a /etc/message/message.txt
sudo chown 2000:3000 /etc/message/message.txt
sudo chmod 640 /etc/message/message.txt
```

On the controller, create a pod to read the message.txt file and print the message to the log.

```
vi my-securitycontext-pod.yml
```

Content of the YAML File

```
apiVersion: v1
kind: Pod
metadata:
  name: my-securitycontext-pod
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', "cat /message/message.txt && sleep 3600"]
    volumeMounts:
    - name: message-volume
      mountPath: /message
  volumes:
  - name: message-volume
    hostPath:
      path: /etc/message
```

Check the pod's log to see the message from the file:

```
kubectl logs my-securitycontext-pod
```

Delete the pod and re-create it, this time with a `securityContext` set to use a user and group that do not have access to the file.

```
kubectl delete pod my-securitycontext-pod --now
```

```
apiVersion: v1
kind: Pod
metadata:
  name: my-securitycontext-pod
spec:
  securityContext:
    runAsUser: 2001
    fsGroup: 3001
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', "cat /message/message.txt && sleep 3600"]
    volumeMounts:
    - name: message-volume
      mountPath: /message
  volumes:
  - name: message-volume
    hostPath:
      path: /etc/message
```

Check the log again. You should see a "permission denied" message.

```
kubectl logs my-securitycontext-pod
```

Delete the pod and re-create it again, this time with a user and group that are able to access the file.

```
kubectl delete pod my-securitycontext-pod --now
```

```
apiVersion: v1
kind: Pod
metadata:
  name: my-securitycontext-pod
spec:
  securityContext:
    runAsUser: 2000
    fsGroup: 3000
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', "cat /message/message.txt && sleep 3600"]
    volumeMounts:
    - name: message-volume
      mountPath: /message
  volumes:
  - name: message-volume
    hostPath:
      path: /etc/message
```

Check the log once more. You should see the message from the file.

```
kubectl logs my-securitycontext-pod
```