Lesson 05 Demo 03

Working with Kubernetes Security Context

Objective: To demonstrate the process of configuring a Kubernetes security context and validating its settings, followed by gaining shell access to a running container within the cluster

Tools required: kubeadm, kubectl, kubelet, and containerd

Prerequisites: A Kubernetes cluster should already be set up (refer to the steps provided in Lesson 02, Demo 01 for guidance).

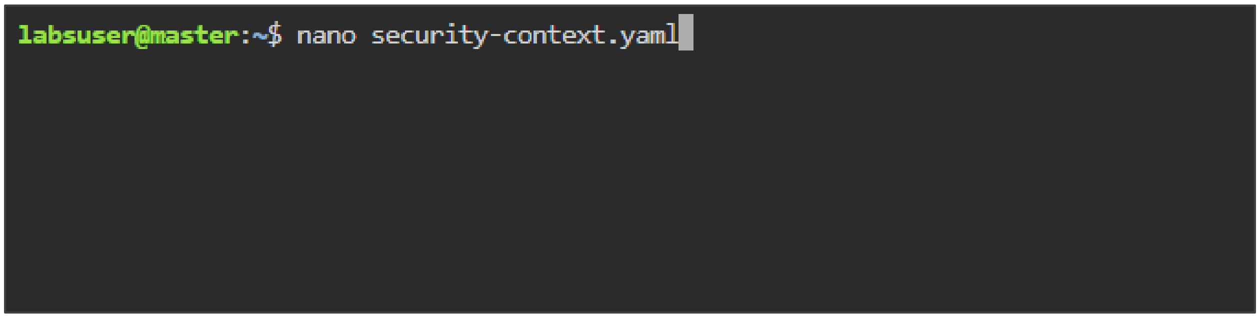
Steps to be followed:

1. Create and verify the security context
2. Access the shell within the running container

Step 1: Create and verify the security context

1.1 Create the YAML file by using the following command:

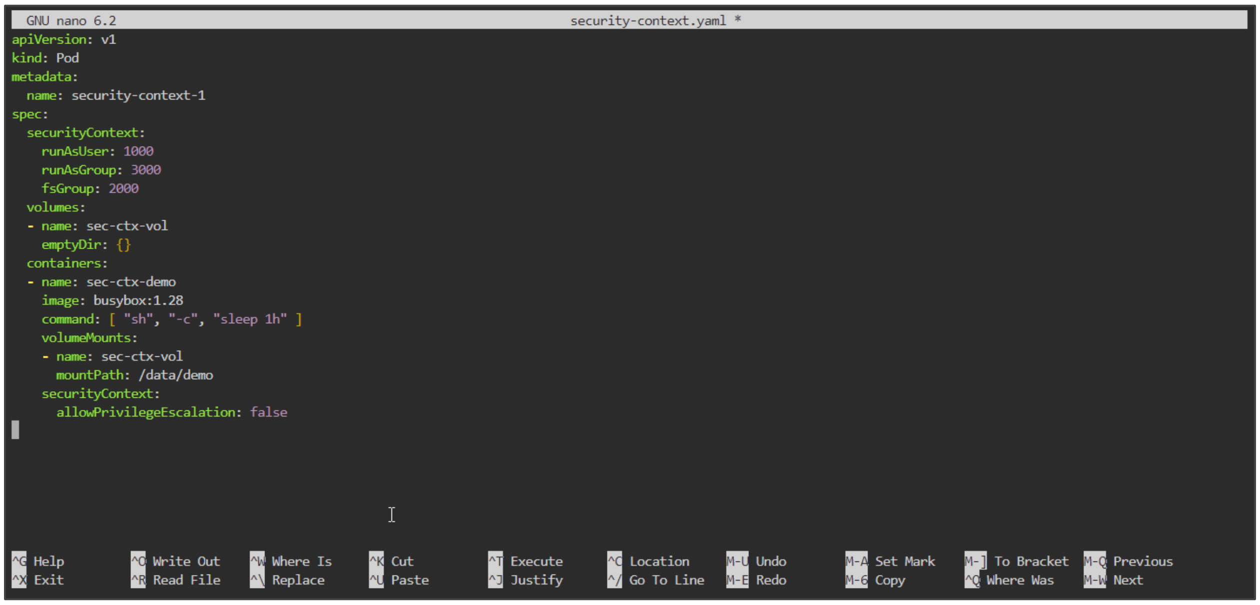
nano security-context.yaml



```
labsuser@master:~$ nano security-context.yaml
```

1.2 Add the following code to the **nano security-context.yaml** file:

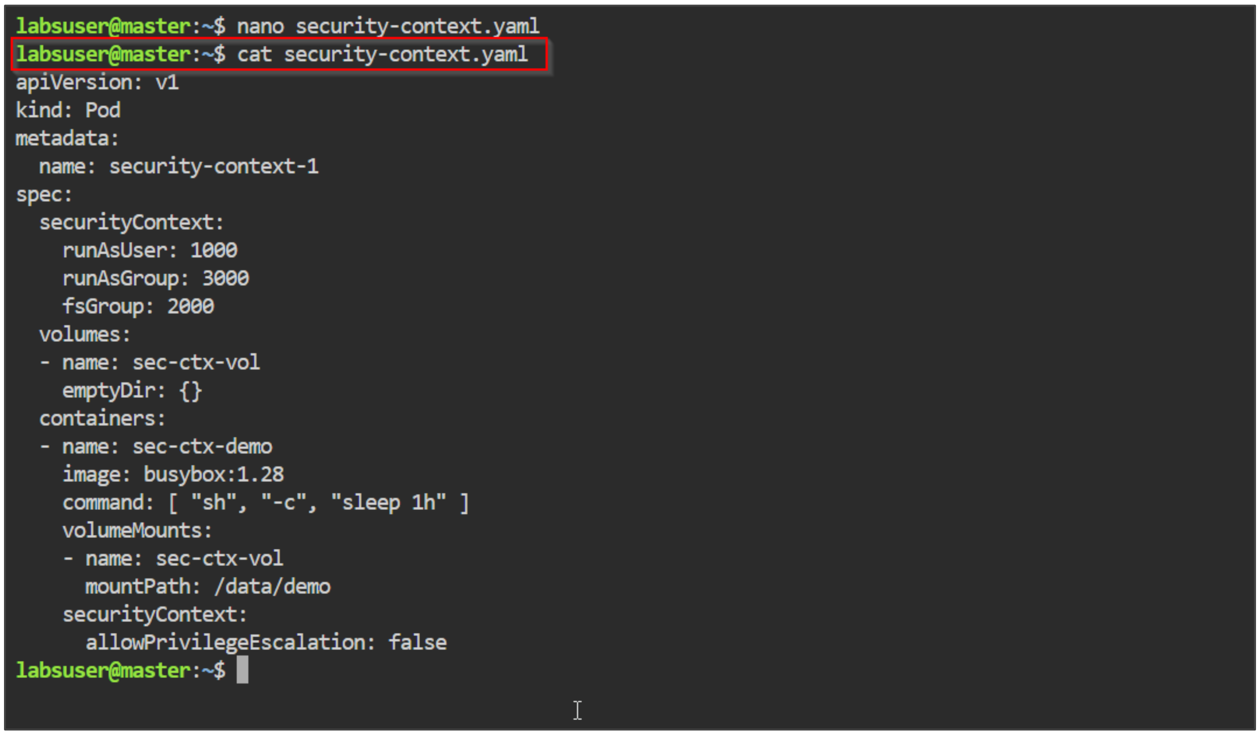
```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-1
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: sec-ctx-vol
    emptyDir: {}
  containers:
  - name: sec-ctx-demo
    image: busybox:1.28
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: sec-ctx-vol
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```



```
  GNU nano 6.2                                   security-context.yaml *
apiVersion: v1
kind: Pod
metadata:
  name: security-context-1
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: sec-ctx-vol
    emptyDir: {}
  containers:
  - name: sec-ctx-demo
    image: busybox:1.28
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: sec-ctx-vol
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false

^G Help      ^O Write Out  ^W Where Is   ^K Cut      ^T Execute   ^C Location   M-U Undo   M-A Set Mark   M-] To Bracket  M-Q Previous
^X Exit      ^R Read File  ^\ Replace    ^U Paste    ^J Justify   ^/ Go To Line M-E Redo   M-6 Copy       ^Q Where Was    M-W Next
```

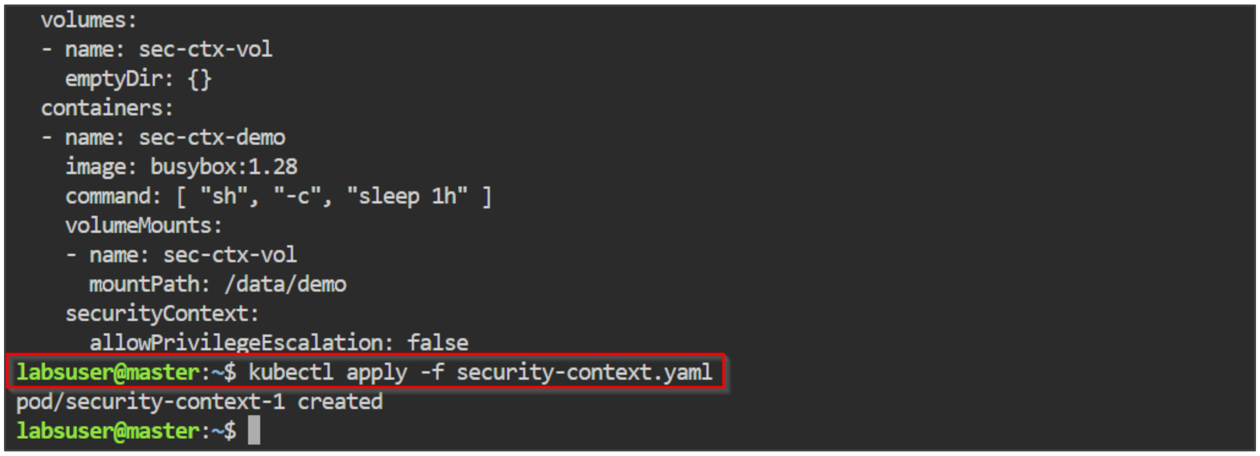
1.3 Use the **cat** command to validate the content of the **nano security-context.yaml** file:



```
labsuser@master:~$ nano security-context.yaml
labsuser@master:~$ cat security-context.yaml
apiVersion: v1
kind: Pod
metadata:
  name: security-context-1
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: sec-ctx-vol
    emptyDir: {}
  containers:
  - name: sec-ctx-demo
    image: busybox:1.28
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: sec-ctx-vol
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
labsuser@master:~$
```

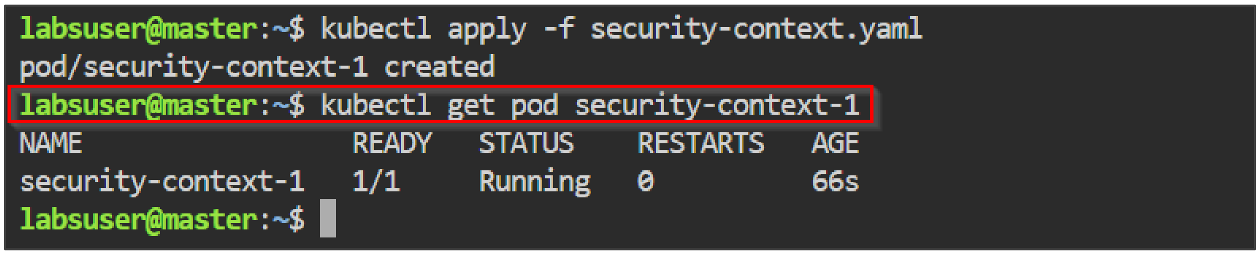
1.4 Create the security context resource by using the following command:

kubectl apply -f security-context.yaml



```
  volumes:
  - name: sec-ctx-vol
    emptyDir: {}
  containers:
  - name: sec-ctx-demo
    image: busybox:1.28
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: sec-ctx-vol
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
labsuser@master:~$ kubectl apply -f security-context.yaml
pod/security-context-1 created
labsuser@master:~$
```

1.5 Verify the **security-context** pod by using the following command:

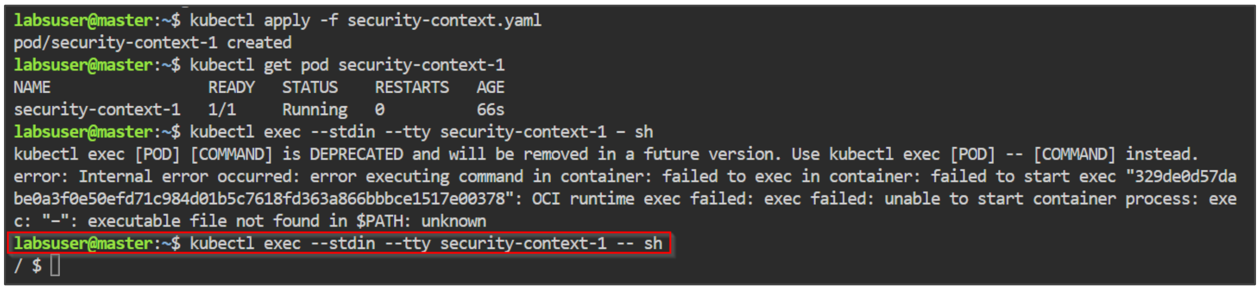


```
labsuser@master:~$ kubectl apply -f security-context.yaml
pod/security-context-1 created
labsuser@master:~$ kubectl get pod security-context-1
NAME                 READY   STATUS    RESTARTS   AGE
security-context-1   1/1     Running   0          66s
labsuser@master:~$
```

Step 2: Access the shell within the running container

2.1 Obtain shell access to the running container by using the following command:

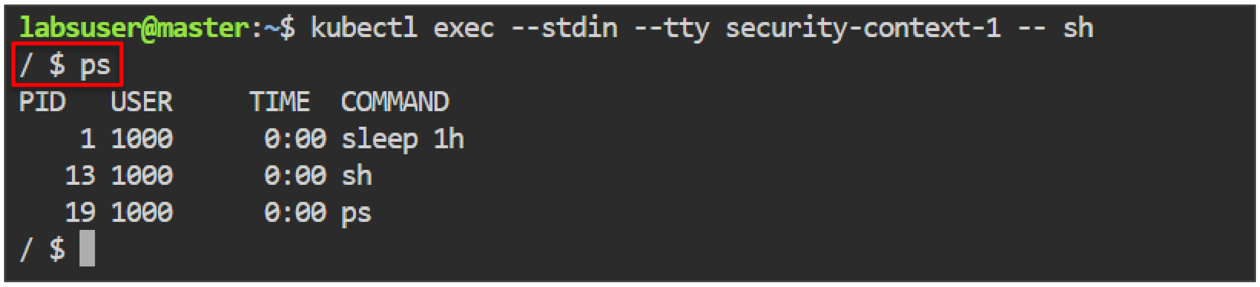
kubectl exec --stdin --tty security-context-1 -- sh



```
labsuser@master:~$ kubectl apply -f security-context.yaml
pod/security-context-1 created
labsuser@master:~$ kubectl get pod security-context-1
NAME                 READY   STATUS    RESTARTS   AGE
security-context-1   1/1     Running   0          66s
labsuser@master:~$ kubectl exec --stdin --tty security-context-1 – sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
error: Internal error occurred: error executing command in container: failed to exec in container: failed to start exec "329de0d57da
be0a3f0e50efd71c984d01b5c7618fd363a866bbbce1517e00378": OCI runtime exec failed: exec failed: unable to start container process: exe
c: "–": executable file not found in $PATH: unknown
labsuser@master:~$ kubectl exec --stdin --tty security-context-1 -- sh
/ $
```

2.2 Use the following command to list the running processes:

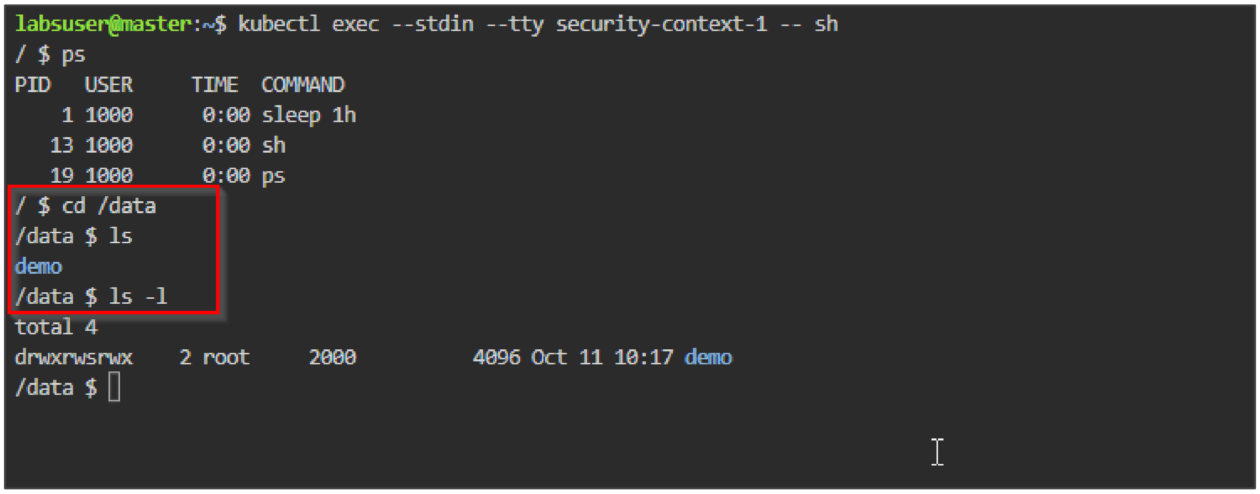
ps



```
labsuser@master:~$ kubectl exec --stdin --tty security-context-1 -- sh
/ $ ps
PID   USER     TIME  COMMAND
    1 1000      0:00 sleep 1h
   13 1000      0:00 sh
   19 1000      0:00 ps
/ $
```

2.3 Navigate **to /data** folder and list the contents by using the following commands:

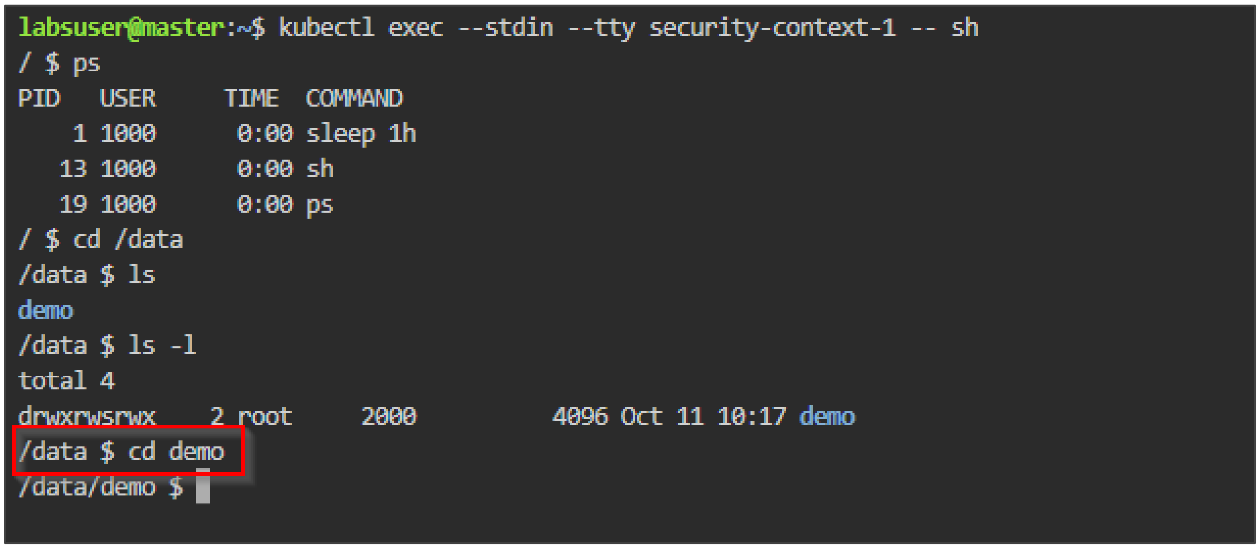
cd /data
ls
ls-l



```
labsuser@master:~$ kubectl exec --stdin --tty security-context-1 -- sh
/ $ ps
PID   USER     TIME  COMMAND
    1 1000      0:00 sleep 1h
   13 1000      0:00 sh
   19 1000      0:00 ps
/ $ cd /data
/data $ ls
demo
/data $ ls -l
total 4
drwxrwsrwx    2 root     2000          4096 Oct 11 10:17 demo
/data $ 
```

2.4 Navigate to the **/data/demo** folder by using the following command:

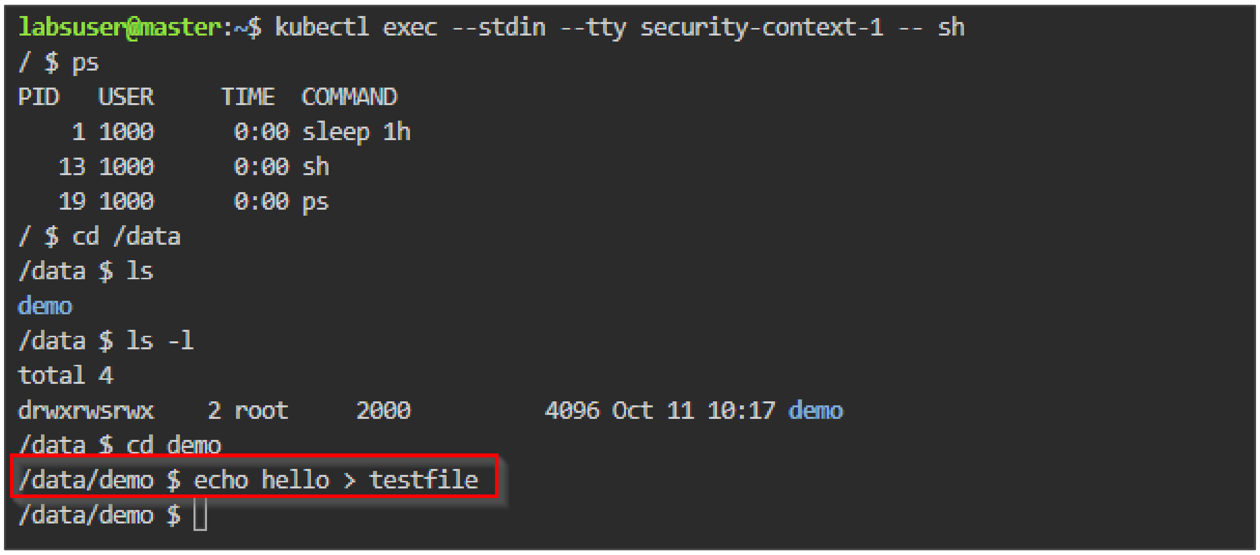
cd demo



```
labsuser@master:~$ kubectl exec --stdin --tty security-context-1 -- sh
/ $ ps
PID   USER     TIME  COMMAND
    1 1000      0:00 sleep 1h
   13 1000      0:00 sh
   19 1000      0:00 ps
/ $ cd /data
/data $ ls
demo
/data $ ls -l
total 4
drwxrwsrwx    2 root     2000          4096 Oct 11 10:17 demo
/data $ cd demo
/data/demo $ 
```

2.5 Create a file by using the following command:

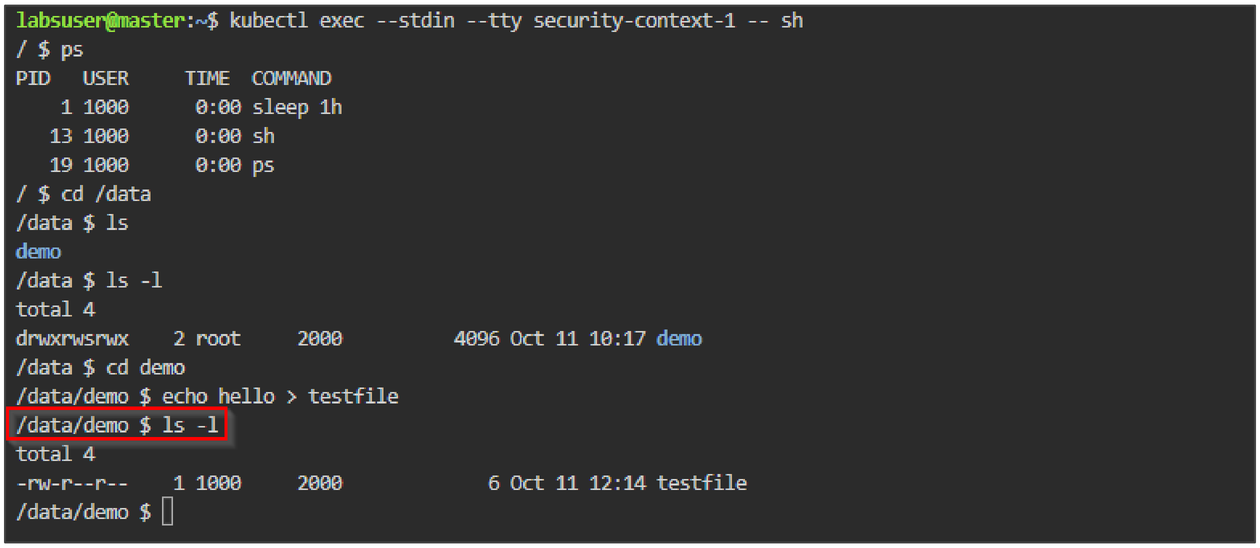
echo hello > testfile



```
labsuser@master:~$ kubectl exec --stdin --tty security-context-1 -- sh
/ $ ps
PID   USER     TIME  COMMAND
    1 1000      0:00 sleep 1h
   13 1000      0:00 sh
   19 1000      0:00 ps
/ $ cd /data
/data $ ls
demo
/data $ ls -l
total 4
drwxrwsrwx    2 root     2000          4096 Oct 11 10:17 demo
/data $ cd demo
/data/demo $ echo hello > testfile
/data/demo $ 
```

2.6 List the files in the **/data/demo** directory by using the following command:

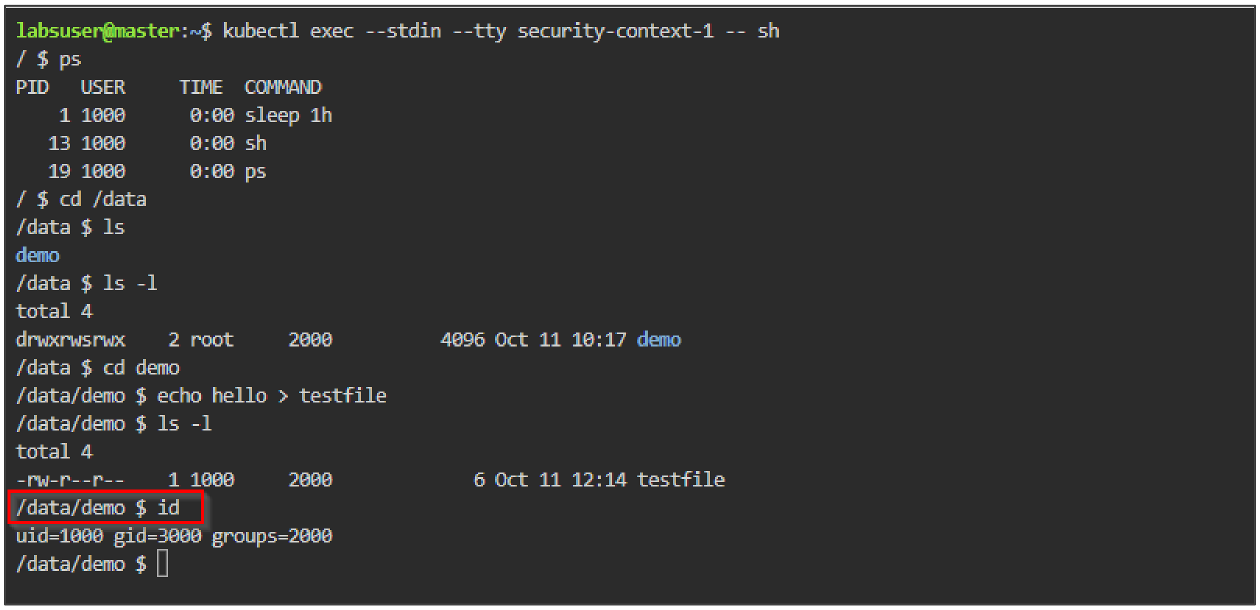
ls -l



```
labsuser@master:~$ kubectl exec --stdin --tty security-context-1 -- sh
/ $ ps
PID   USER     TIME  COMMAND
    1 1000      0:00 sleep 1h
   13 1000      0:00 sh
   19 1000      0:00 ps
/ $ cd /data
/data $ ls
demo
/data $ ls -l
total 4
drwxrwsrwx    2 root     2000          4096 Oct 11 10:17 demo
/data $ cd demo
/data/demo $ echo hello > testfile
/data/demo $ ls -l
total 4
-rw-r--r--    1 1000     2000             6 Oct 11 12:14 testfile
/data/demo $ 
```

2.7 Execute the following command to get the respective user and group ID:

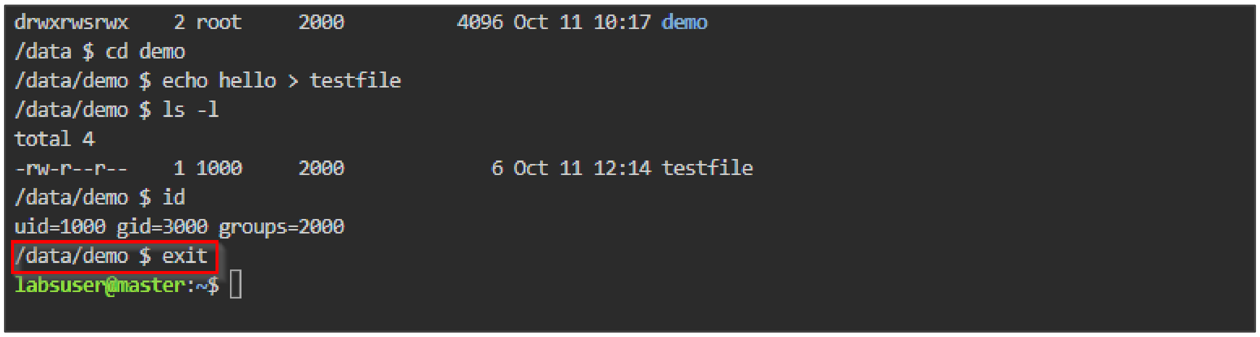
id



```
labsuser@master:~$ kubectl exec --stdin --tty security-context-1 -- sh
/ $ ps
PID   USER     TIME  COMMAND
    1 1000      0:00 sleep 1h
   13 1000      0:00 sh
   19 1000      0:00 ps
/ $ cd /data
/data $ ls
demo
/data $ ls -l
total 4
drwxrwsrwx    2 root     2000          4096 Oct 11 10:17 demo
/data $ cd demo
/data/demo $ echo hello > testfile
/data/demo $ ls -l
total 4
-rw-r--r--    1 1000     2000             6 Oct 11 12:14 testfile
/data/demo $ id
uid=1000 gid=3000 groups=2000
/data/demo $ 
```

2.8 Exit the shell by using the following command:

exit



```
drwxrwsrwx    2 root     2000          4096 Oct 11 10:17 demo
/data $ cd demo
/data/demo $ echo hello > testfile
/data/demo $ ls -l
total 4
-rw-r--r--    1 1000     2000             6 Oct 11 12:14 testfile
/data/demo $ id
uid=1000 gid=3000 groups=2000
/data/demo $ exit
labsuser@master:~$ 
```

By following these steps, you have successfully set up a Kubernetes security context, verified its configuration, and accessed a running container shell.