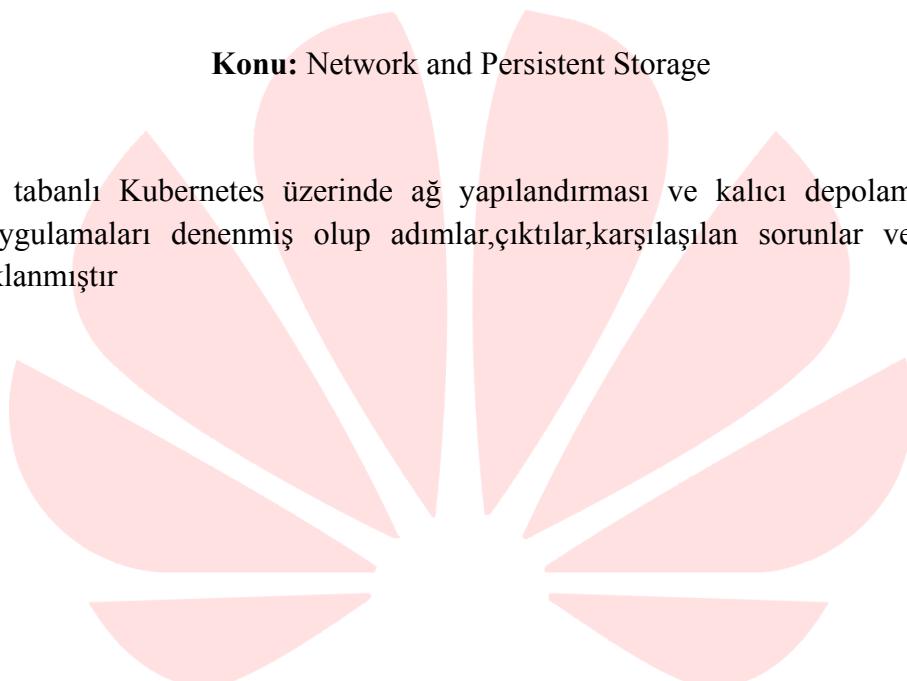


Huawei Kubernetes CCE Bootcamp Lab-2

Uygulama Raporu

Konu: Network and Persistent Storage

Özet: CCE tabanlı Kubernetes üzerinde ağ yapılandırması ve kalıcı depolama(persistent storage) uygulamaları denenmiş olup adımlar, çıktılar, karşılaşılan sorunlar ve çözümleri raporda açıklanmıştır



Hazırlayan
Mustafa ESEN

1) Giriş

Bu laboratuvar çalışmasında, Huawei Cloud Container Engine (CCE) ortamı üzerinde Kubernetes ağ yapılandırması ve kalıcı depolama (Persistent Storage) uygulamaları gerçekleştirilmiştir. Çalışma kapsamında Kubernetes Network ve Service yapıları incelenmiş, kalıcı veri depolama için Persistent Volume bileşenleri kullanılmış ve durum bilgisine sahip uygulamaların yönetimi amacıyla StatefulSet kavramı uygulanmıştır. Laboratuvarın temel amacı, Kubernetes üzerinde çalışan uygulamaların ağ ve depolama bileşenleriyle birlikte nasıl yönetildiğini pratik olarak deneyimlemek ve gerçek ortam senaryolarına yönelik bilgi ve tecrübe kazanmaktır.

2) Donanım ve Çevresel Bilgiler

Platform: Huawei Cloud • KooLabs Sandbox

İşletim Sistemi: CentOS (ECS üzerinde)

Kubernetes Cluster: cce01

Namespace: default / kube-system

CLI Aracı: kubectl v1.27+

Kullanıcı: root@ecs-k8s

3) Lab Ortamına Giriş ve Aktifleştirme

KooLabs'te açılan sanal masaüstü ortamında tarayıcıdan cloud hesabına sistem tarafından belirlenmiş SandBox kullanıcı bilgileri ile IAM User tipi ile giriş yapıldı. Daha sonra pratikler yapılırken gerekli olacak cluster, node, ecs temini için aşağıdaki adımlar uygulandı

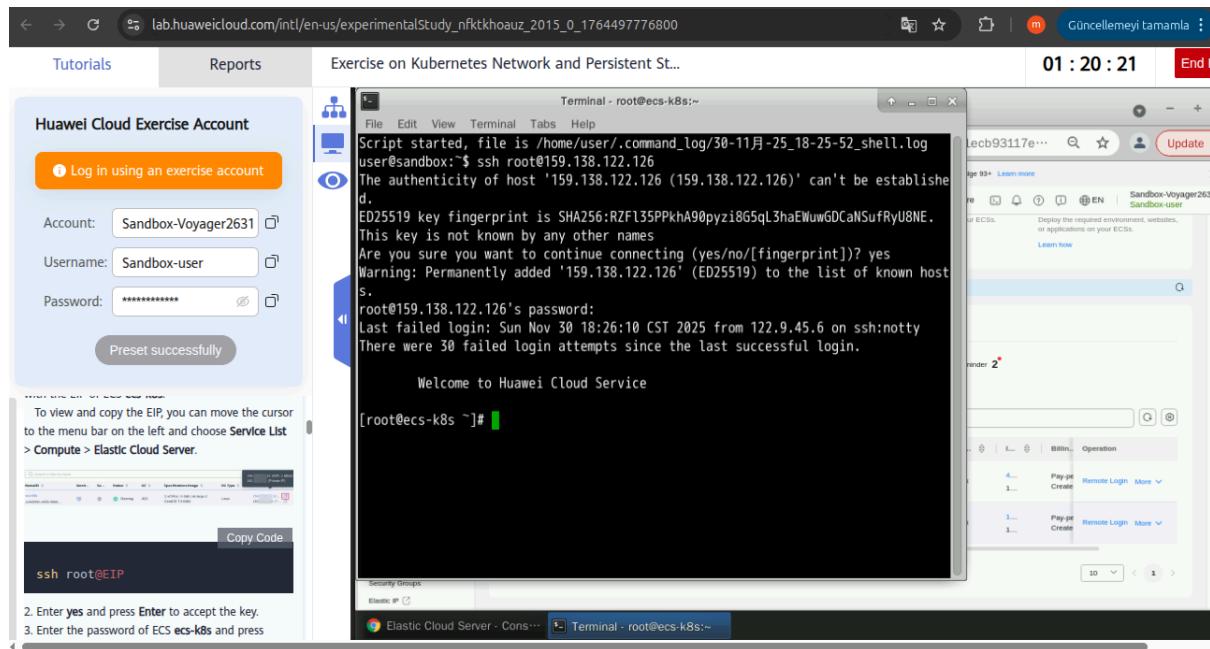
→ Yönergede de belirtildiği üzere yine yönergede belirtildiği gibi **cce01** adında bir cluster oluşturuldu.

→ Bu **cluster'a** yine yönergede belirtilen özelliklerde **node-cce01** adında bir node oluşturuldu.

→ Bu aşamada elastic cloud server lab ortamında alınmış ve atanmış şekilde gelmektedir.

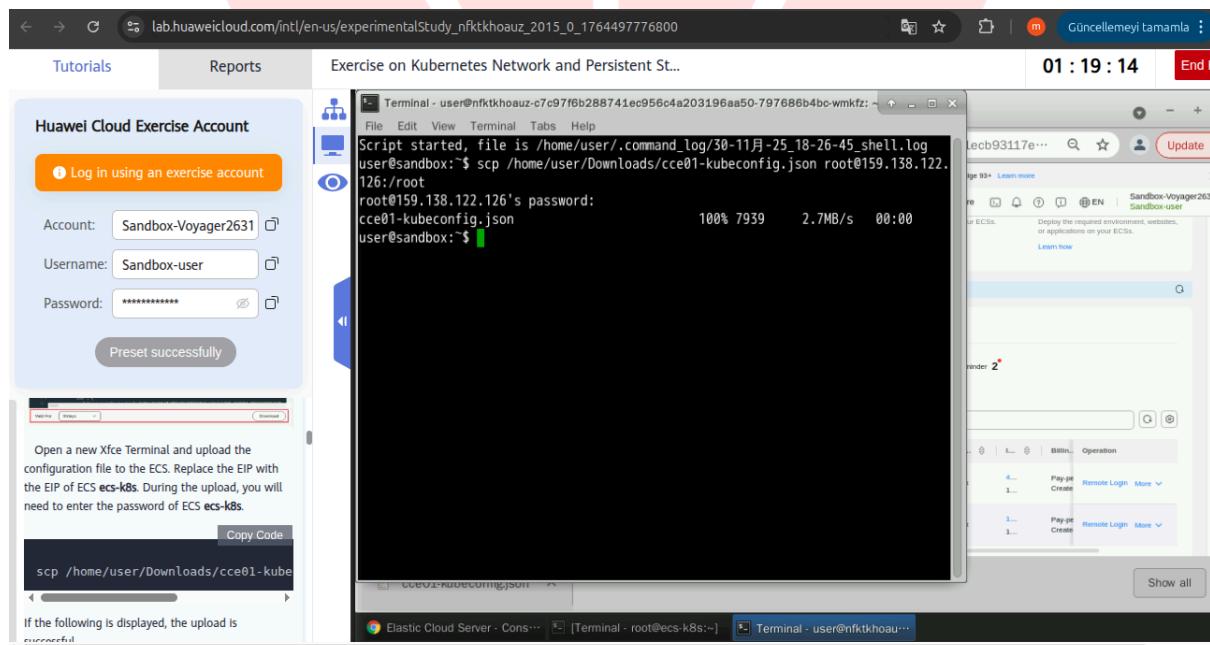
4) Makine Bağlantısı ve kubeconfig Dosyası Import Etme

Lab ortamındaki ecs sunucusuna public eip ve root şifre bilgileri ile SSH bağlantısı kuruldu. "Welcome to Huawei Cloud Service" mesajı ile işlem doğrulandı.



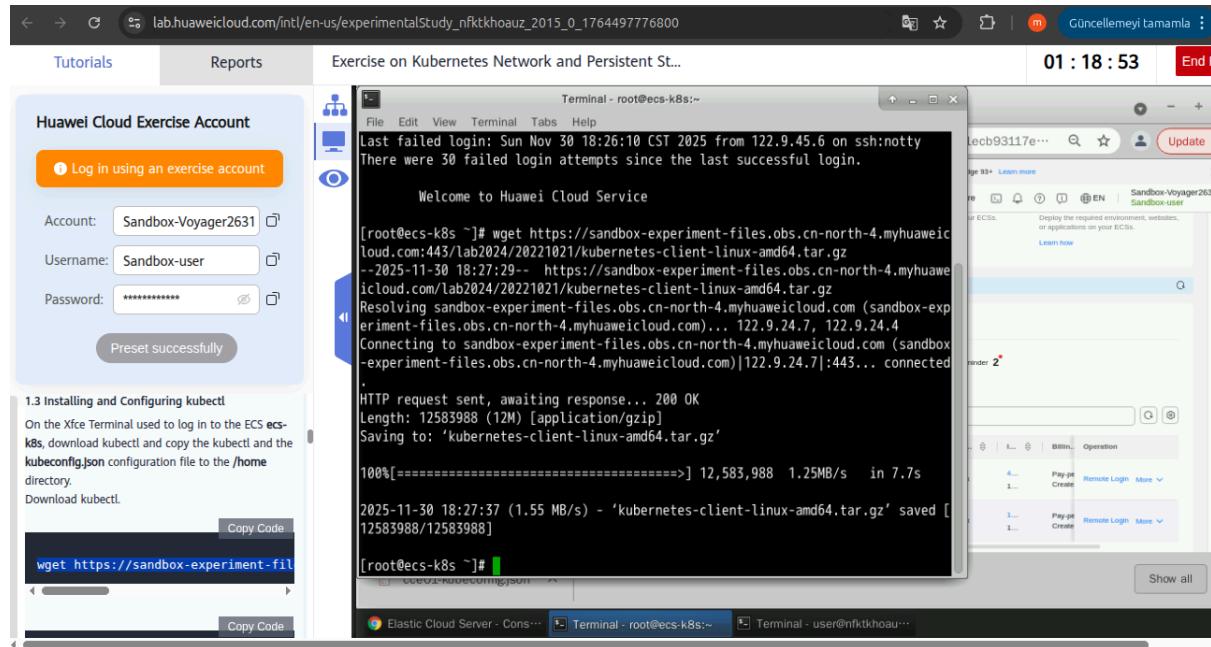
(Görsel 1.0: SSH bağlantısı.)

Daha sonra Kubernetes cluster'a erişim için gerekli olan config dosyası lab ortamı local'inden ecs sunucusuna kopyalanıp kubectl ile ecs yönetimini sağlayacak hale getirilmiştir.



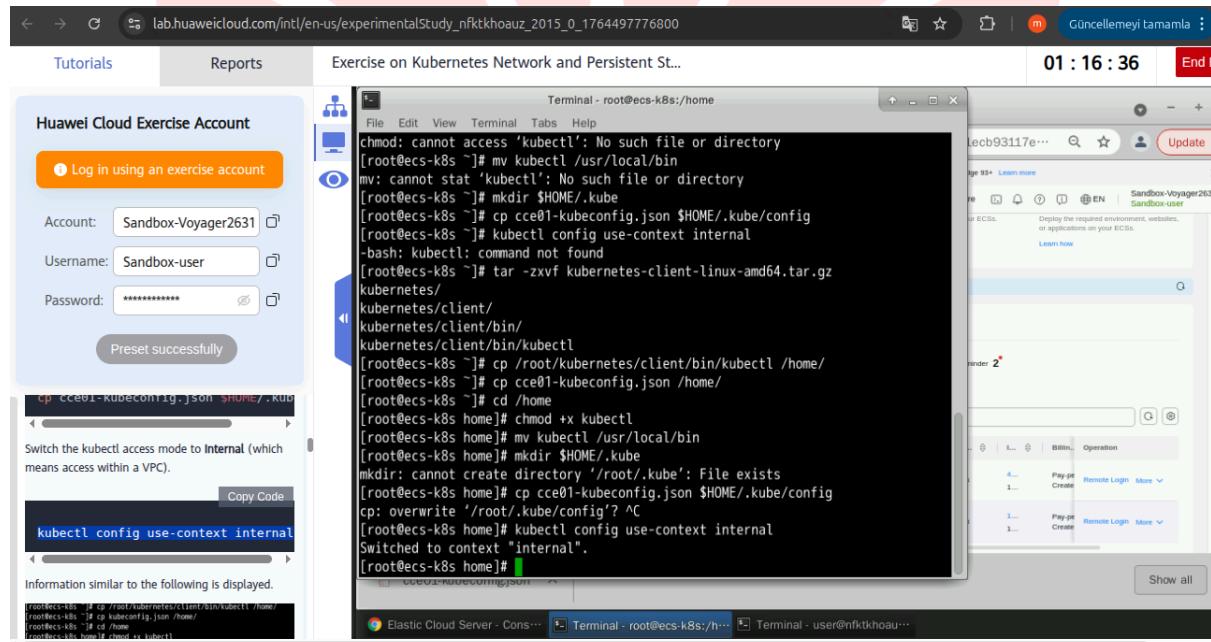
(Görsel 1.1: Kubeconfig dosyası ecs yükleme.)

Gerekli kubectl istemcisi OBS üzerinden wget komutuyla ECS'ye indirilmiş ve paket kaydedilmiştir.



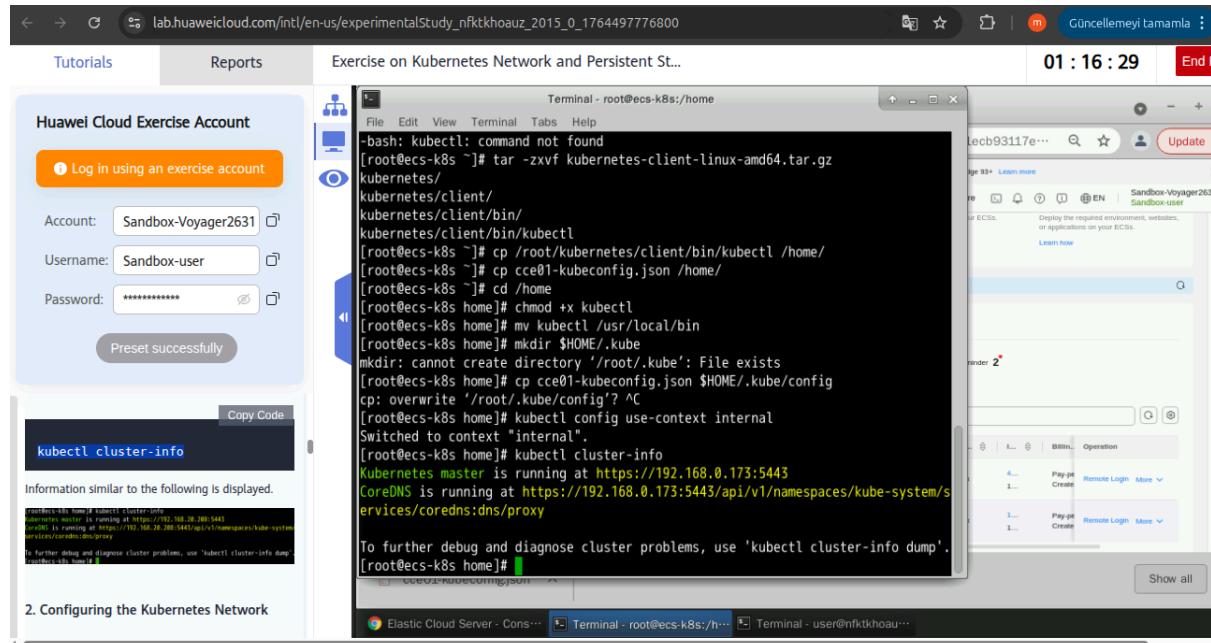
(Görsel 1.2: Kubectl istemcisini ECS'ye kurma.)

Kubectl kurulumu tamamlanmış, kubeconfig dosyası yapılandırılmış ve CCE cluster için internal context başarıyla aktif edilmiştir.



(Görsel 1.3: Kubectl kurulumu tamamlanması.)

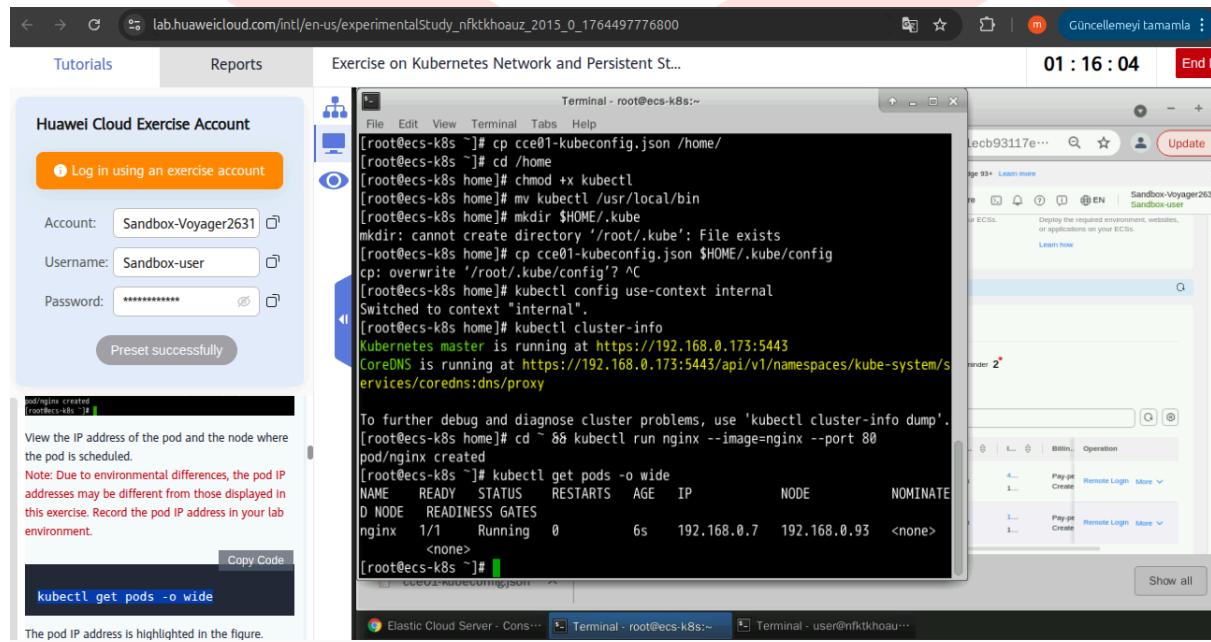
Son olarak **kubectl cluster-info** komutu ile önceden tamamlanan kurulum kontrol edilmiş, Kubernetes Servisi'nin aktif ve sağlıklı olduğu gözlemlenmiştir.



(Görsel 1.4: Kurulum sonrası kontrol.)

2) Pod Oluşturma ve Çalışma Durumunun Doğrulanması

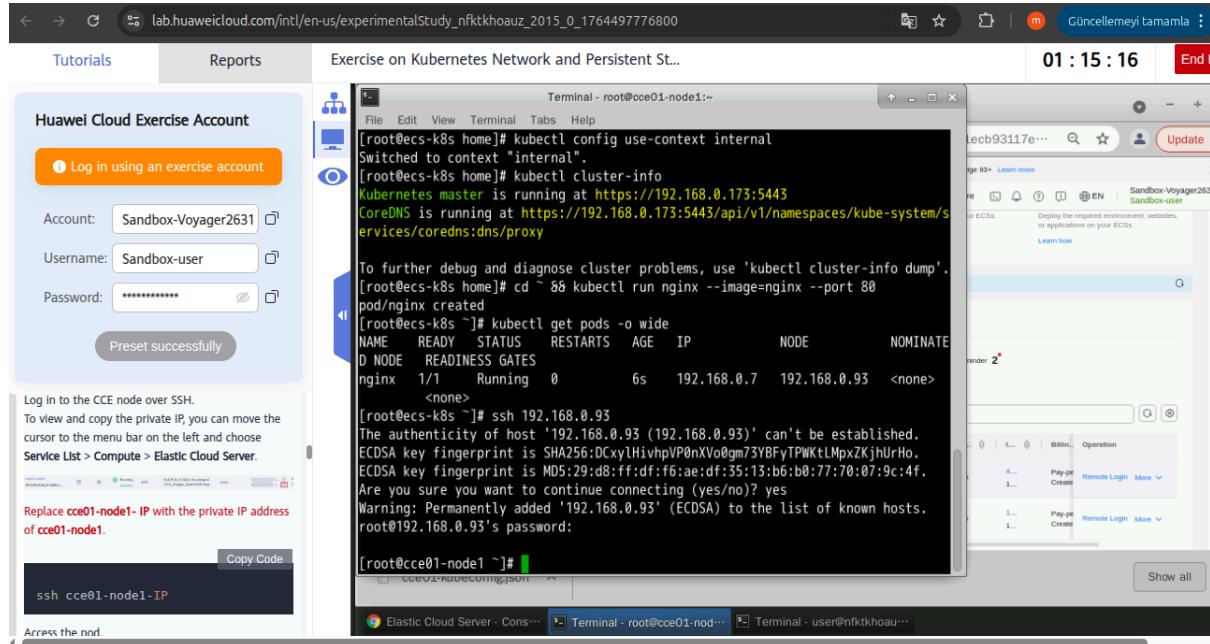
Bu aşamada test amacıyla nginx pod'u oluşturulmuştur. **kubectl run** komutuyla pod başarıyla başlatılmış ardından **kubectl get pods -o wide** komutu ile çalışma durumu atanan IP adresi ve üzerinde çalıştığı node bilgileri kontrol edilmiştir. Bu kontroller sonrasında cluster'in pod çalıştırma ve ağ yapılandırması doğrulanmıştır.



(Görsel 1.5: Nginx pod'u oluşturma ve kontrolü.)

3) Node Üzerinden SSH Bağlantısı

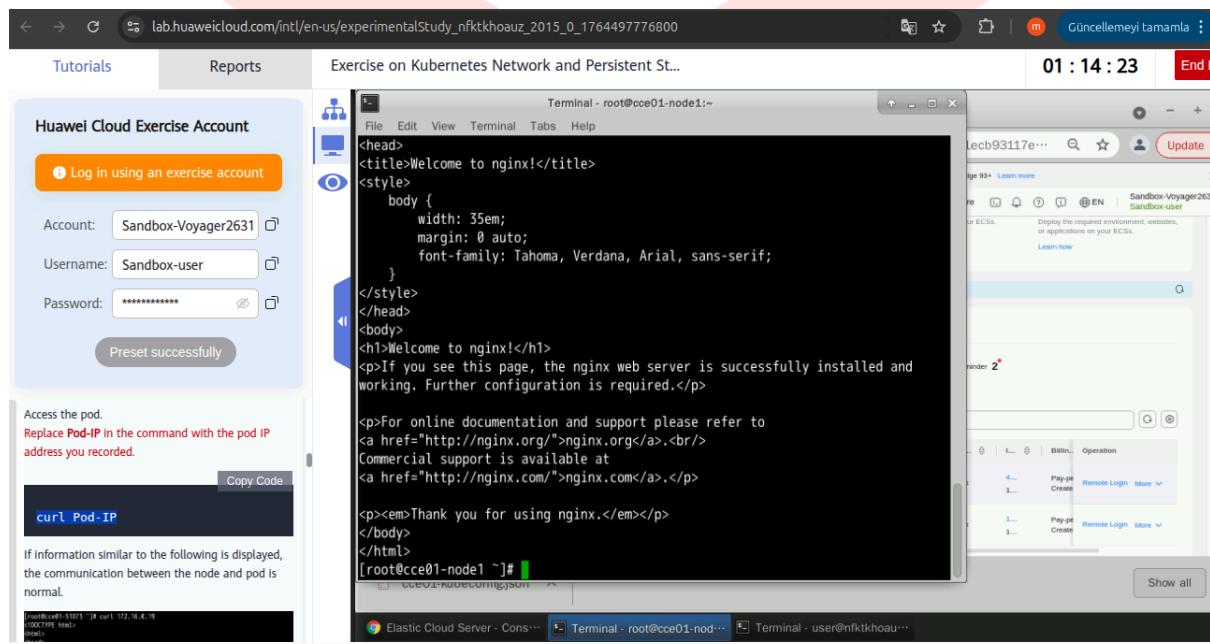
Oluşturulan nginx pod'un çalıştığı node'a özel IP adresi kullanılarak **ssh 192.168.0.93** komutu ile node'a erişim sağlanmıştır.



(Görsel 1.6: Nginx pod'unun çalıştığı node'a ssh bağlantısı ile bağlanma.)

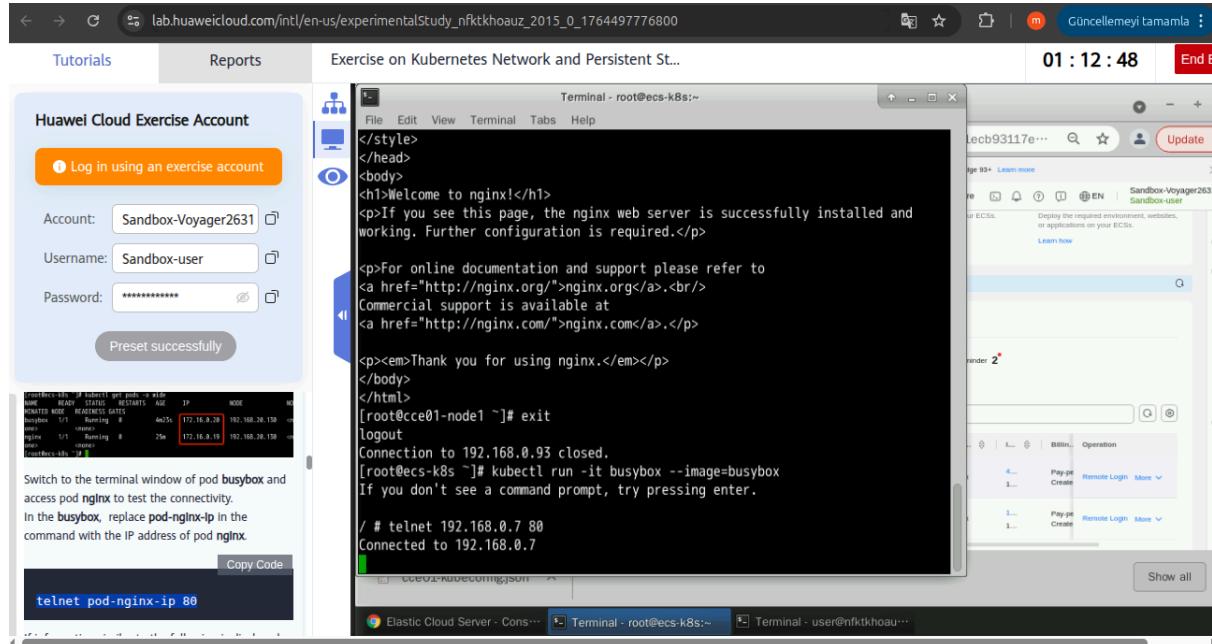
5) Pod Erişiminin Doğrulanması

Bu aşamada bir önceki aşamada bağlanılan node üzerinden pod IP ile curl<podip> komutu ile pod'a erişilerek pod'un çalıştığı doğrulanmıştır.



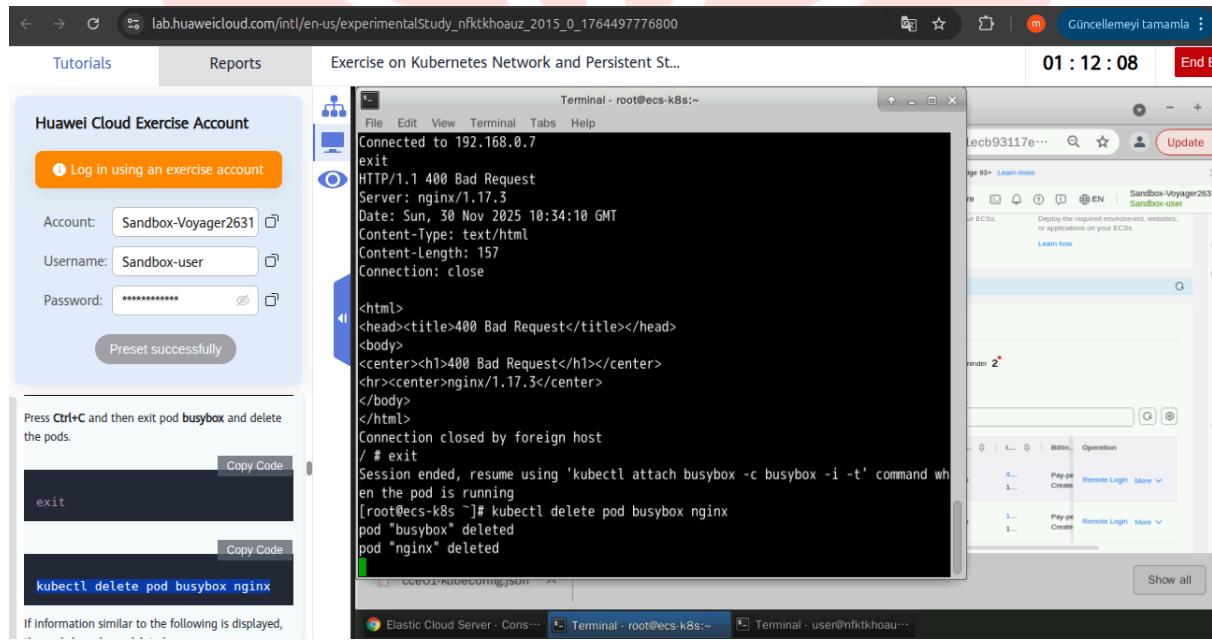
(Görsel 1.7 : Pod doğrulanması.)

Daha sonra pod erişimi test edilmesi amacıyla **kubectl run -it busybox --image busybox** komutu ile pod başlatılmış ardından busybox içerisindeki **nginx** pod'unun ip adresi ve **80** portu hedeflenerek telnet atılmıştır(**telnet 192.168.07 80**). Başarılı bağlantı sonrası podlar arası iletişimini sağlıklı olduğu gözlemlenmiştir.



(Görsel 1.8: Busybox üzerinden kontrol.)

Son olarak ise test amaçlı oluşturulan busybox ve nginx pod'ları **kubectl delete pod busybox nginx** komutu ile silinmiştir.

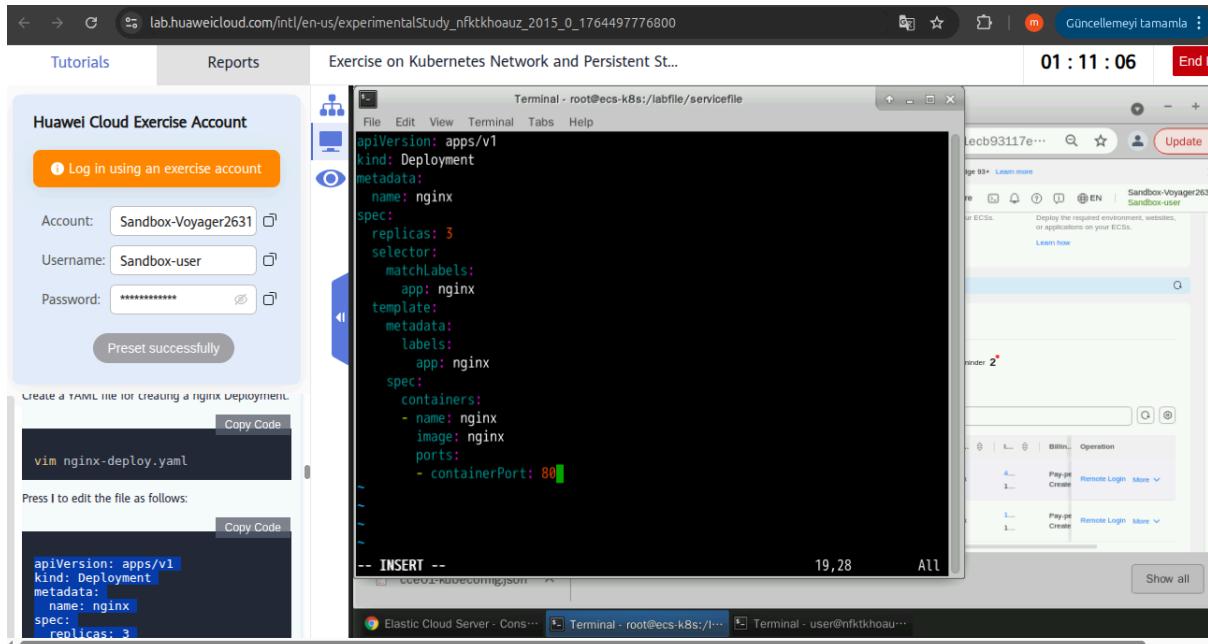


(Görsel 1.9: Busybox ve nginx pod'ları silinmesi.)

6) Nginx Uygulaması Yönetimi

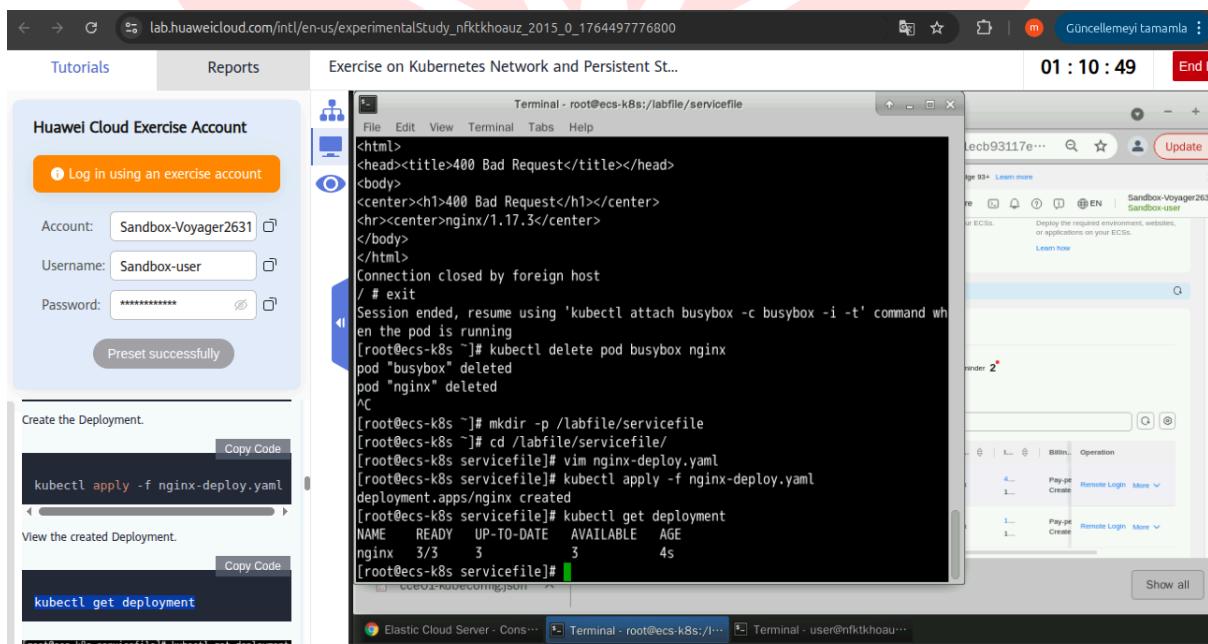
6.1) Deployment

İlk olarak nginx uygulaması için kubernetes deployment tanımı hazırlanmış. YAML dosyasında adı, replica sayısı(3), container image bilgisi ve 80 numaralı pod tanımlanmıştır.



(Görsel 2.0: Nginx Deployment tanımı YAML dosyası.)

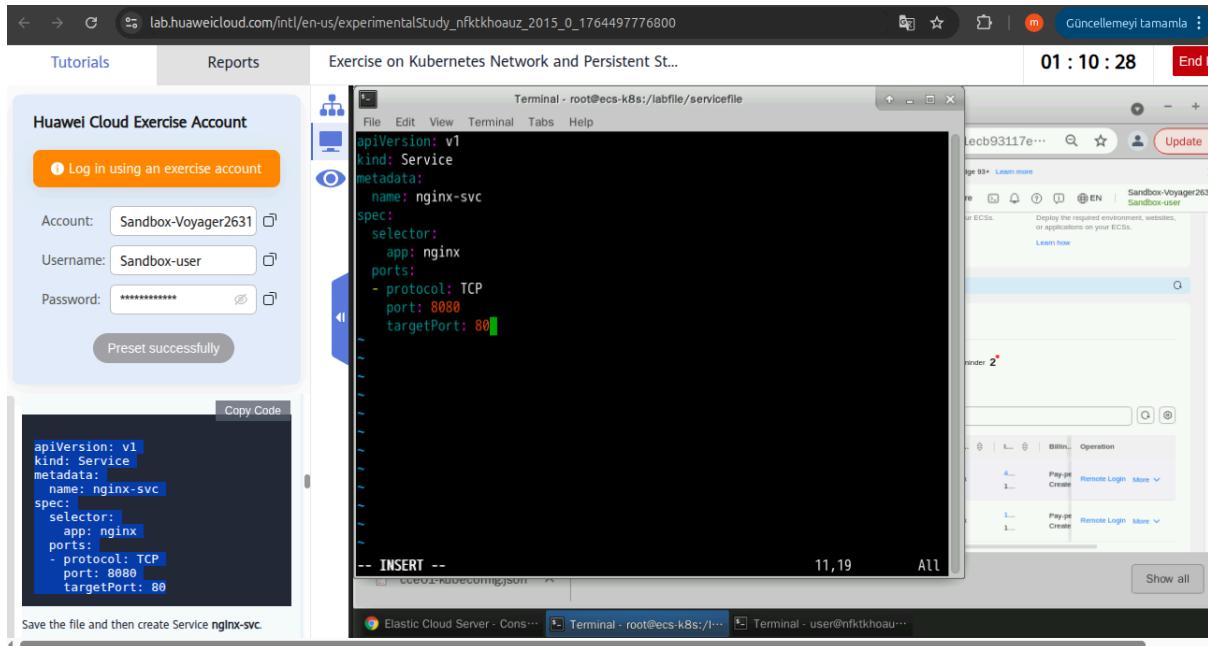
Sonraki aşamada bu YAML dosyası **kubectl apply -f nginx-deployment.yaml** komutu ile cluster'a uygulanmış sonuç ise **kubectl get deployments** komutu ile kontrol edildiğinde sağlıklı çalıştığı gözlemlenmiştir.



(Görsel 2.1: Deploy etme ve kontrol.)

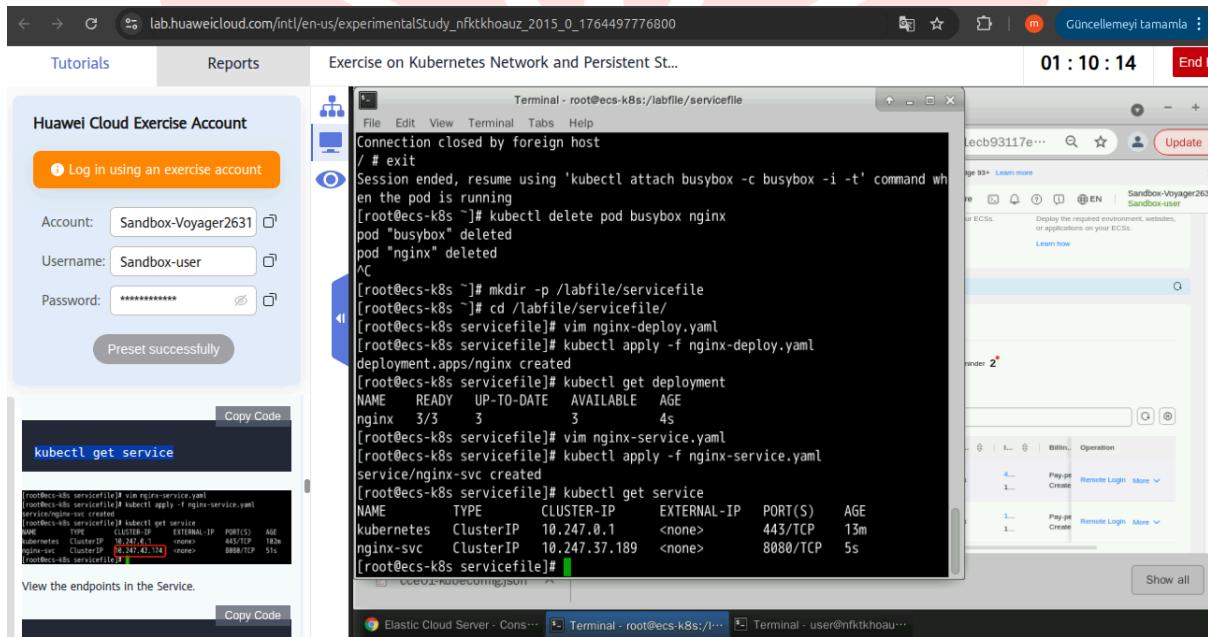
6.2) Service

Oluşturulan nginx deployment'a erişim için kubernetes service tanımı yapılmıştır. Yapılandırmada **app:nginx** etiketine sahip podlar seçilmiş ve **8080** üzerinden gelen trafik **80 portuna** yönlendirilmiştir.



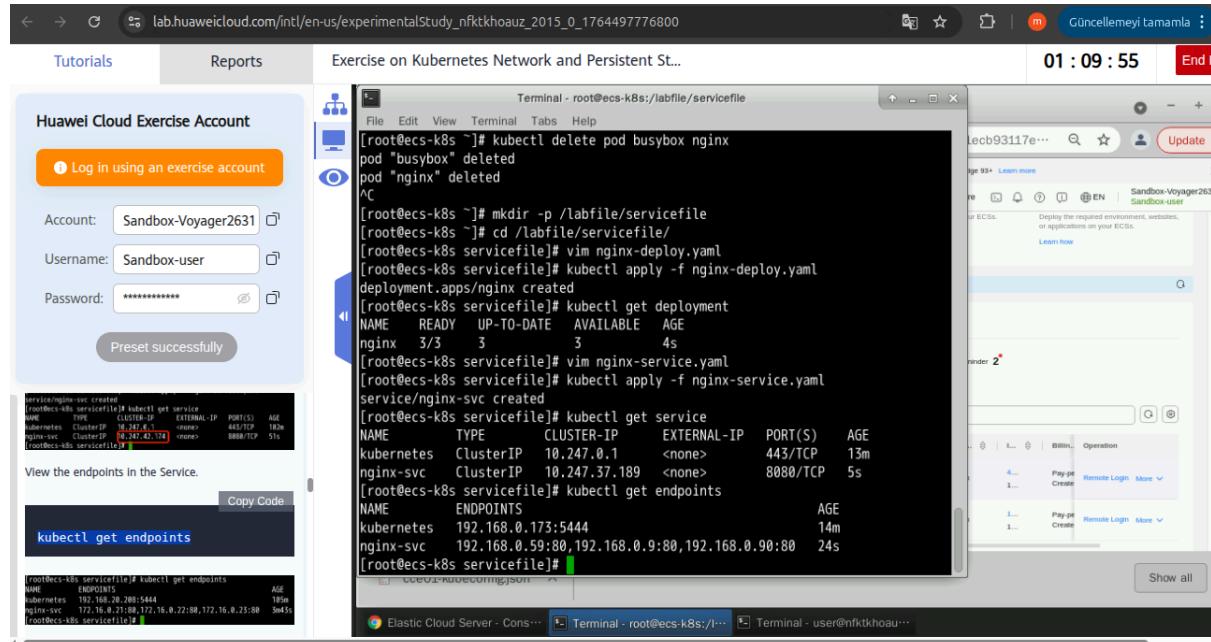
(Görsel 2.2: Nginx Service Tanımı.)

Sonrasında hazırlanan service tanımı **kubectl apply -f nginx-service.yaml** komutu ile cluster'a uygulanmış sonuç ise **kubectl get service** komutu ile kontrol edilmiştir.



(Görsel 2.3: Service tanımının uygulanması ve kontrolü.)

Son olarak nginx service'in hangi pod'lara trafik yönlendirdiğini doğrulamak amacıyla **kubectl get endpoints** komutu çalıştırılmış çıktıya göre service'in birden fazla pod ip ve 80 portu ile listelenmesi üzere service ile pod'lar arasındaki iş yükü ve dağılım doğrulanmıştır.



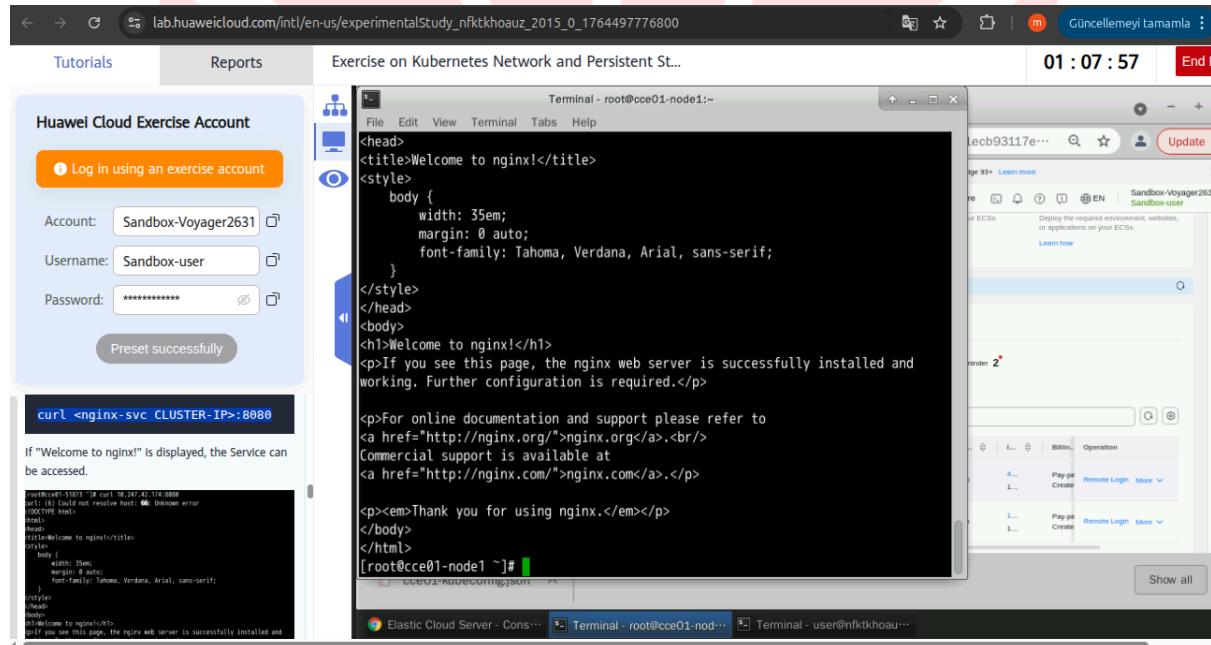
The screenshot shows the Huawei Cloud Exercise Account interface. On the left, there's a sidebar with 'Huawei Cloud Exercise Account' and fields for 'Account' (Sandbox-Voyager2631), 'Username' (Sandbox-user), and 'Password'. Below these are buttons for 'Log in using an exercise account' and 'Preset successfully'. In the center, a terminal window titled 'Terminal - root@ecs-k8s:/labfile/servicefile' shows the command history for creating an nginx service and checking its endpoints:

```
[root@ecs-k8s ~]# kubectl delete pod busybox nginx
pod "busybox" deleted
[root@ecs-k8s ~]# pod "nginx" deleted
^C
[root@ecs-k8s ~]# mkdir -p /labfile/servicefile
[root@ecs-k8s ~]# cd /labfile/servicefile/
[root@ecs-k8s servicefile]# vim nginx-deploy.yaml
[root@ecs-k8s servicefile]# kubectl apply -f nginx-deploy.yaml
deployment.apps/nginx created
[root@ecs-k8s servicefile]# kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx     3/3     3           3           4s
[root@ecs-k8s servicefile]# vim nginx-service.yaml
[root@ecs-k8s servicefile]# kubectl apply -f nginx-service.yaml
service/nginx-svc created
[root@ecs-k8s servicefile]# kubectl get service
NAME      TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.247.0.1    <none>        443/TCP   13m
nginx-svc   ClusterIP   10.247.37.189  <none>        8080/TCP  5s
[root@ecs-k8s servicefile]# kubectl get endpoints
NAME      ENDPOINTS   AGE
kubernetes  192.168.0.173:5443   14m
nginx-svc   192.168.0.59:80,192.168.0.90:80,192.168.0.90:80   24s
[root@ecs-k8s servicefile]#
```

On the right, there's a browser window showing the 'Exercise on Kubernetes Network and Persistent St...' page with a 'Logs' tab. The logs show the deployment of the nginx service and its endpoints.

(Görsel 2.4: Service endpoint kontrolü.)

Daha sonra nginx service'e curl üzerinden erişilerek service'in çalıştığı doğrulanmıştır.



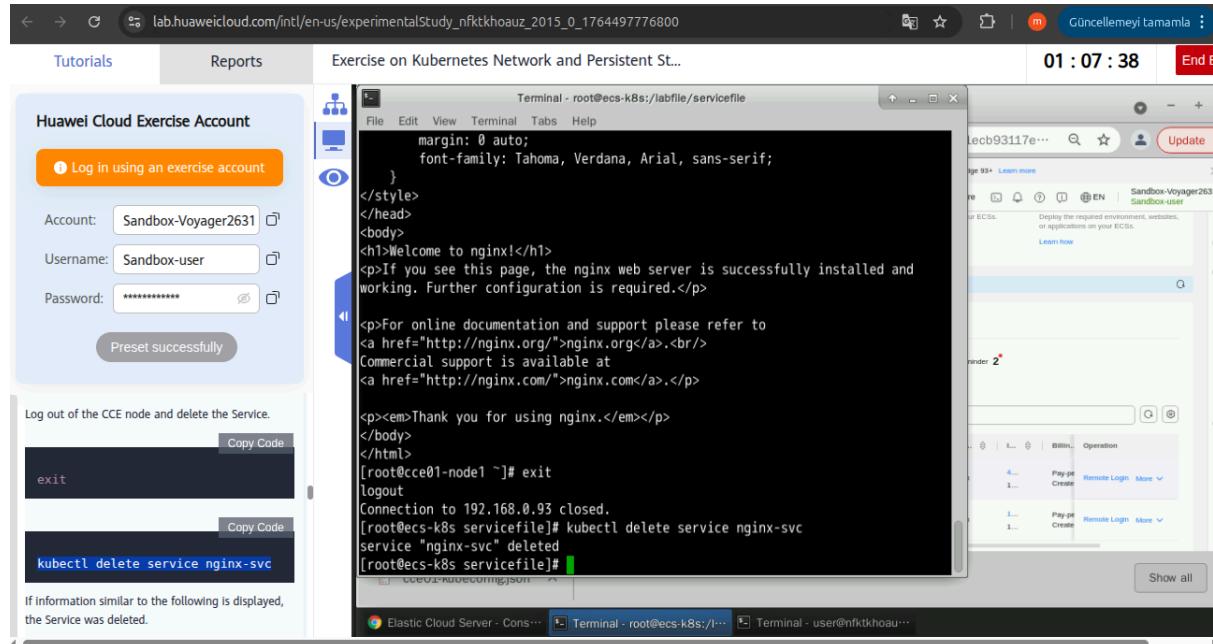
The screenshot shows the Huawei Cloud Exercise Account interface. The central terminal window titled 'Terminal - root@cce01-node1:' shows the curl command output for the nginx service:

```
curl <nginx-svc CLUSTER-IP>:8080
If "Welcome to nginx!" is displayed, the Service can be accessed.
curl: (7) Could not resolve host: nginx-svc: Unknown error
100%[====>] 0.00B/s
```

Below the terminal, a message says 'If "Welcome to nginx!" is displayed, the Service can be accessed.' The browser window on the right shows the 'Logs' tab with the curl command and its output.

(Görsel 2.5: Nginx Service çalışmasını doğrulama.)

En son adımda ise **kubectl delete service nginx-svc** komutu ile bu servis silinerek sistem kaynakları sonraki aşamalar için temizlenmiştir.

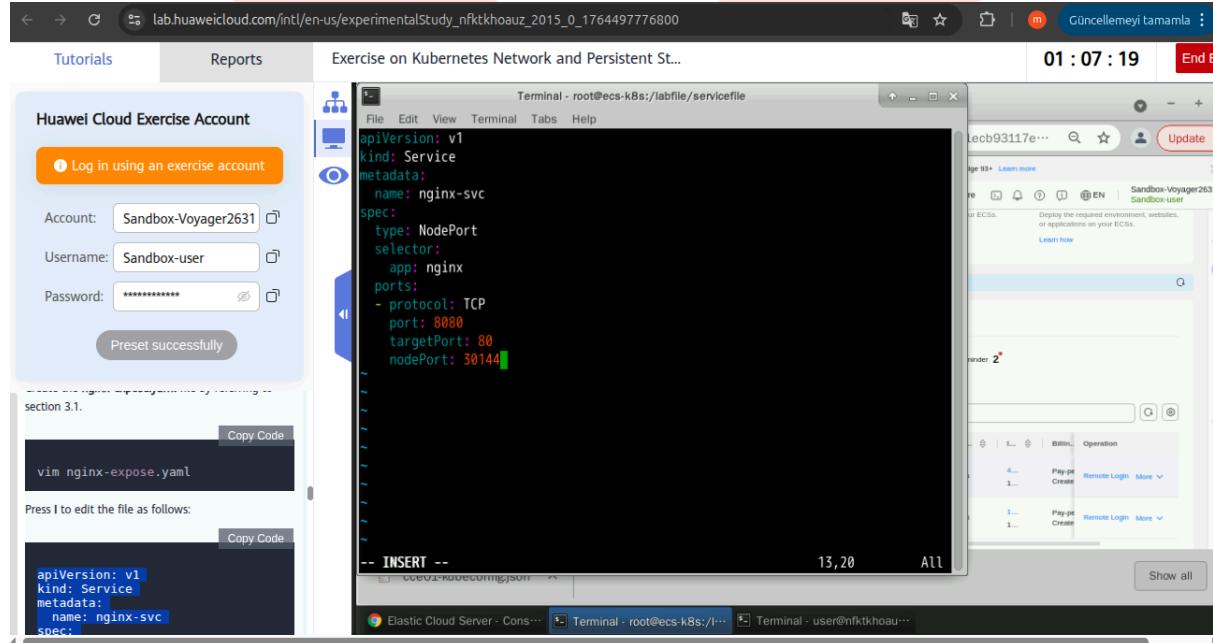


The screenshot shows the Huawei Cloud Exercise Account interface. On the left, there's a form for logging in with account 'Sandbox-Voyager2631', username 'Sandbox-user', and password '*****'. Below it, a message says 'Preset successfully'. In the center, a terminal window shows the command 'kubectl delete service nginx-svc' being run, followed by the output: 'service "nginx-svc" deleted'. To the right, a browser window shows the Nginx default page with the text 'Welcome to nginx!'. A sidebar on the right lists various services and applications.

(Görsel 2.6: Nginx Service silinmesi.)

7) Nginx Service'in Node Port Olarak Yeniden Tanımlanması

Bu aşamada ise nginx service dış erişime açılacak şekilde yeniden tanımlanmıştır. 8080'den gelen trafiği container içindeki 80 portuna yönlendirilmesi ve 30144 port üzerinden dışarıdan erişim sağlanması tanımlanmıştır.



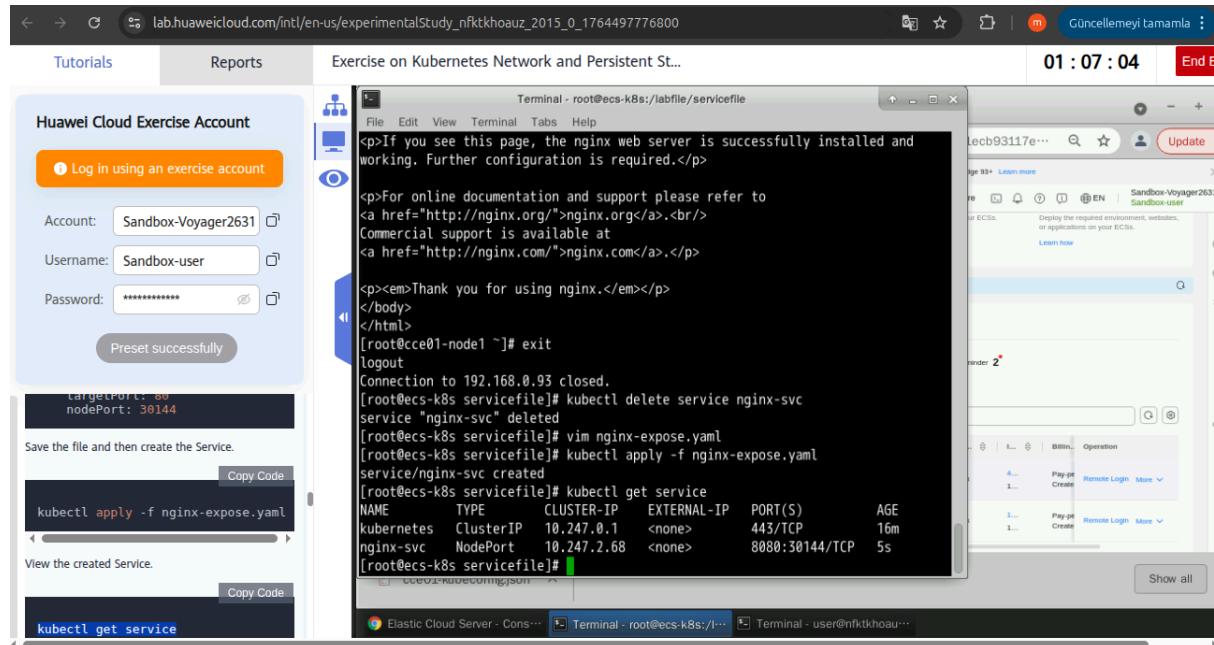
The screenshot shows the Huawei Cloud Exercise Account interface. On the left, there's a form for logging in with account 'Sandbox-Voyager2631', username 'Sandbox-user', and password '*****'. Below it, a message says 'Preset successfully'. In the center, a terminal window shows the command 'vim nginx-expose.yaml' being run. The file content is shown in the terminal:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
      nodePort: 30144
```

 To the right, a browser window shows the Nginx default page with the text 'Welcome to nginx!'. A sidebar on the right lists various services and applications.

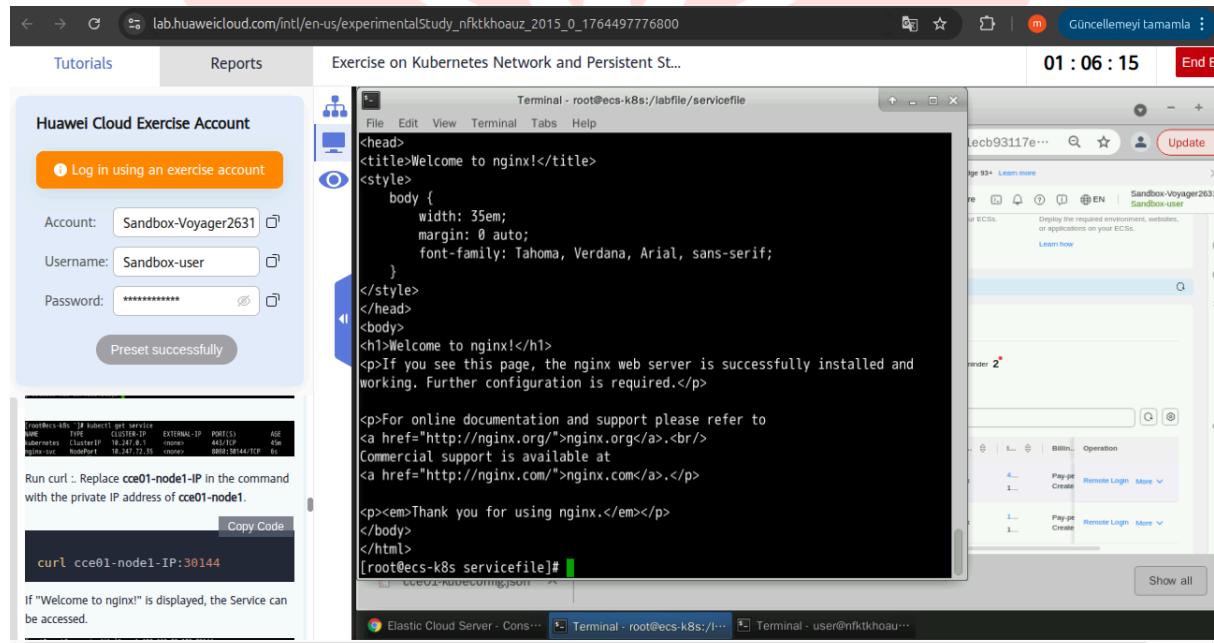
(Görsel 2.7: Nginx Service'in NodePort olarak yeniden tanımlanması.)

Sornaki adımda ise NodePort service **kubectl apply -f nginx-expose.yaml** komutu ile uygulanmış ve **kubectl get service** komutu ile kontrol edilmiştir.



(Görsel 2.8: NodePort service'in uygulanması ve kontrol edilmesi.)

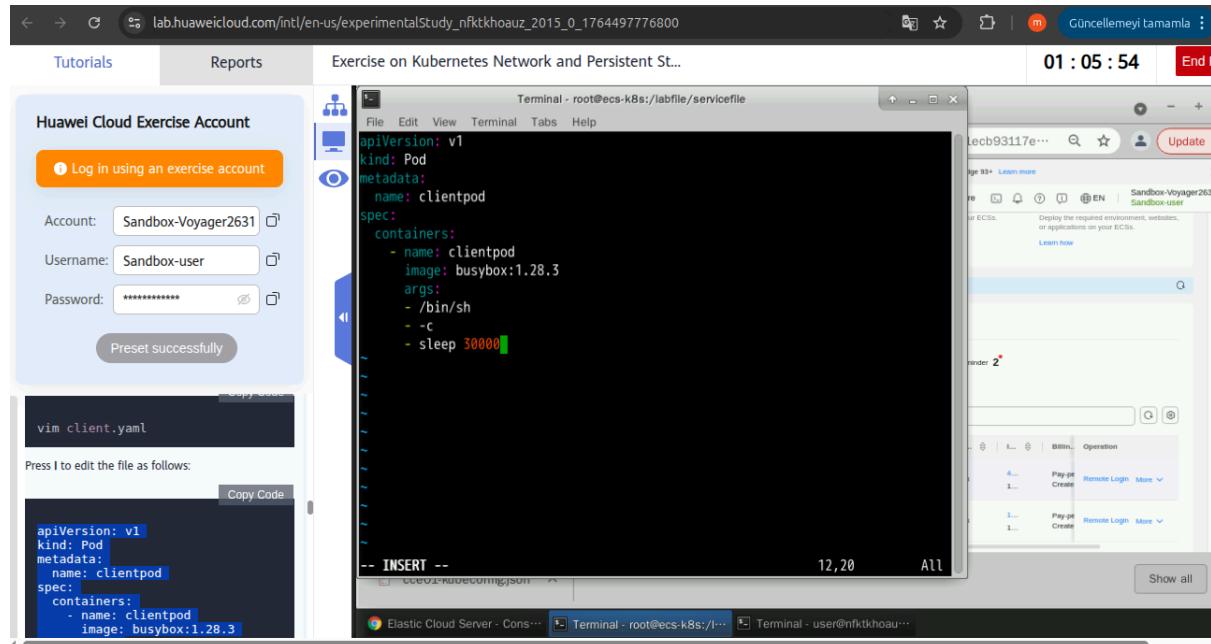
Daha sonra nginx uygulamasına node ip'si ve NodePort kullanılarak (30144) erişim sağlanmış mesaj doğrultusunda çalıştığı ve uygulamanın dışarıdan erişilebilir olduğu gözlemlenmiştir.



(Görsel 2.9: Erişim kontrolü.)

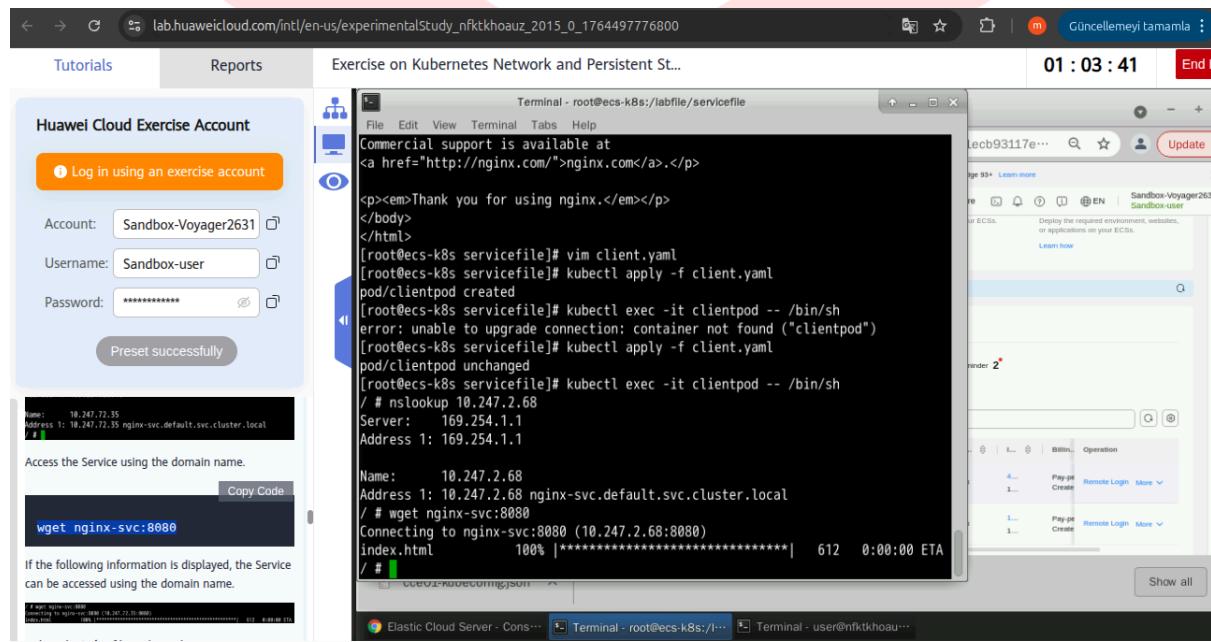
8)ClientPod Oluşturma

Servis ve pod erişimlerinin testleri için BusyBox image kullanan bir client pod tanımlanmıştır.



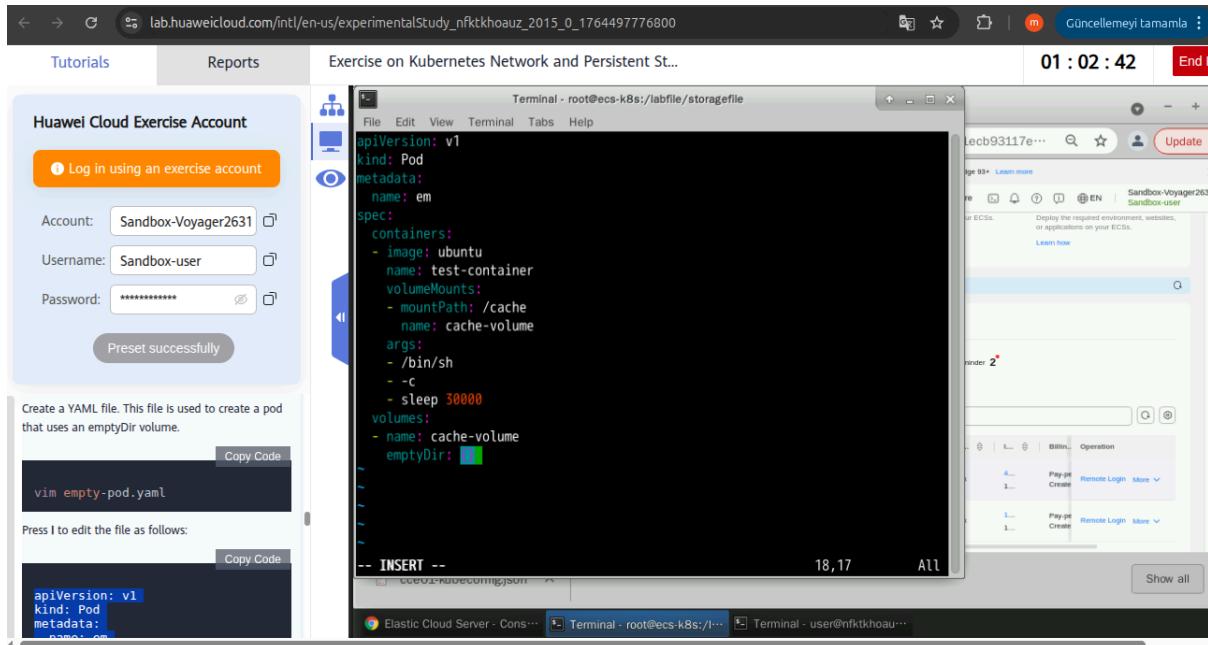
(Görsel 3.0: Client Pod tanımı.)

Daha sonra oluşturulan servis **kubectl apply -f client.yaml** komutu uygulanmış **kubectl exec -it clientpod --/bin/sh** komutu ile nslookup atılarak servisin DNS çözmede başarılı olduğu gözlenmiştir.



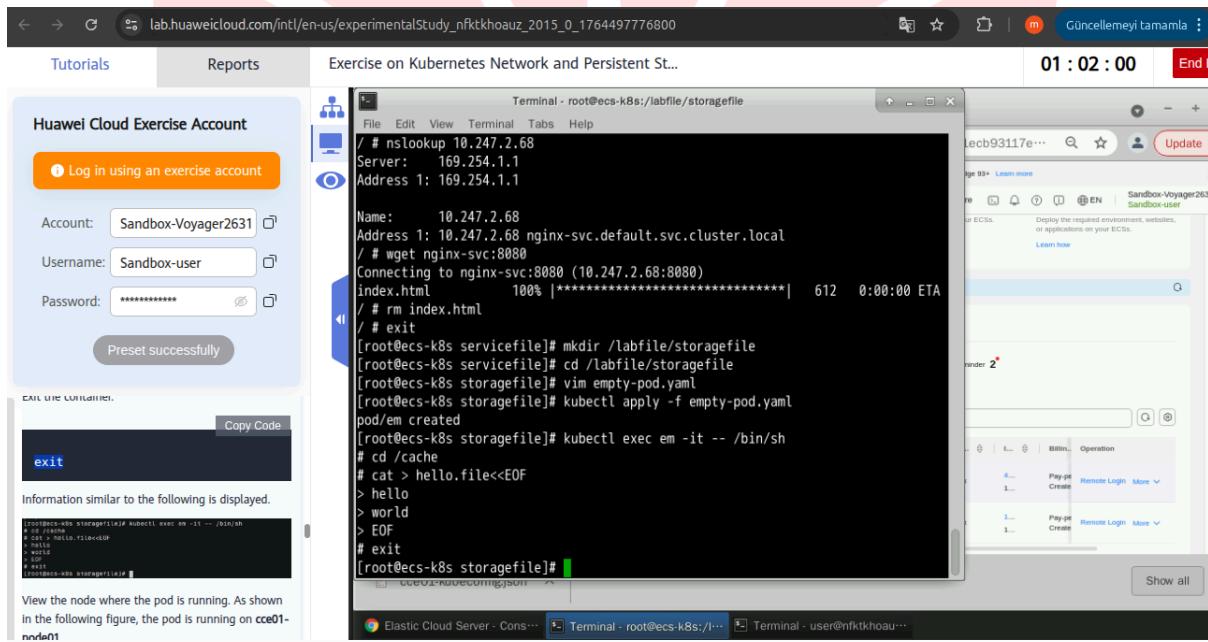
(Görsel 3.1: ClientPod kontrol ve nslookup)

Daha sonra pod çalıştığı sürece geçerli olan silinince kaybolan geçici depolama mekanizmasını test etmek için cache volume adında emptyDir volume dosyası oluşturuldu.



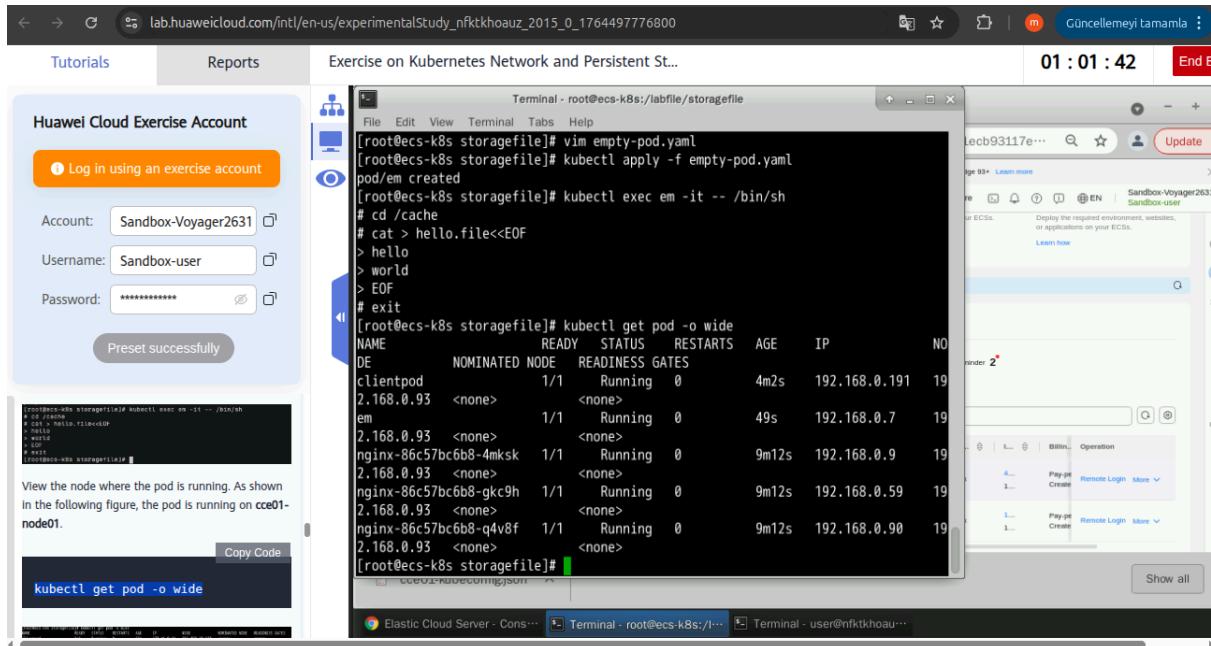
(Görsel 3.2: Cache Volume tanımı.)

Daha sonra bu emptyDir 'i kullanan pod içeresine girilerek test amaçlı hello.file adında dosya oluşturulup içesine veri yazıldı.



(Görsel 3.3: hello.file dosyası.)

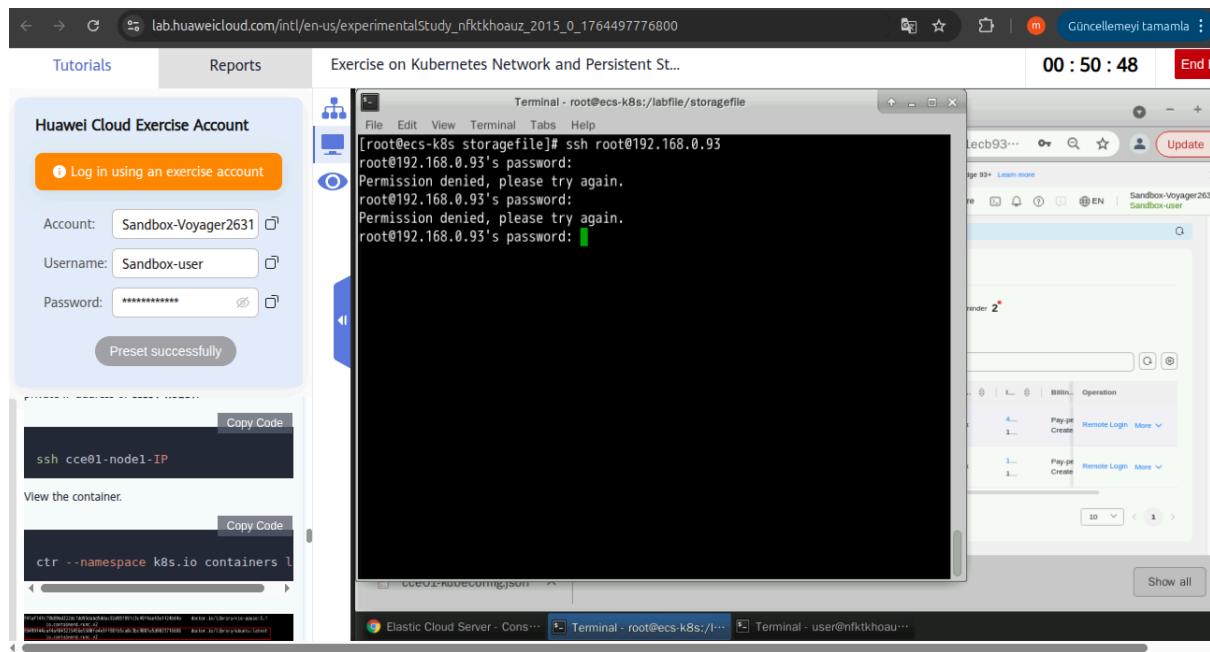
Daha sonra **kubectl get pod -o wide** komutu kullanılarak emptyDir volume kullanan pod'un çalışma durumu, IP adresi ve hangi node üzerinde çalıştığı görüntülenmiştir. Pod'un **Running** durumda olduğu ve belirli bir node (cce01-node1) üzerinde başarıyla çalıştığı doğrulanmıştır.



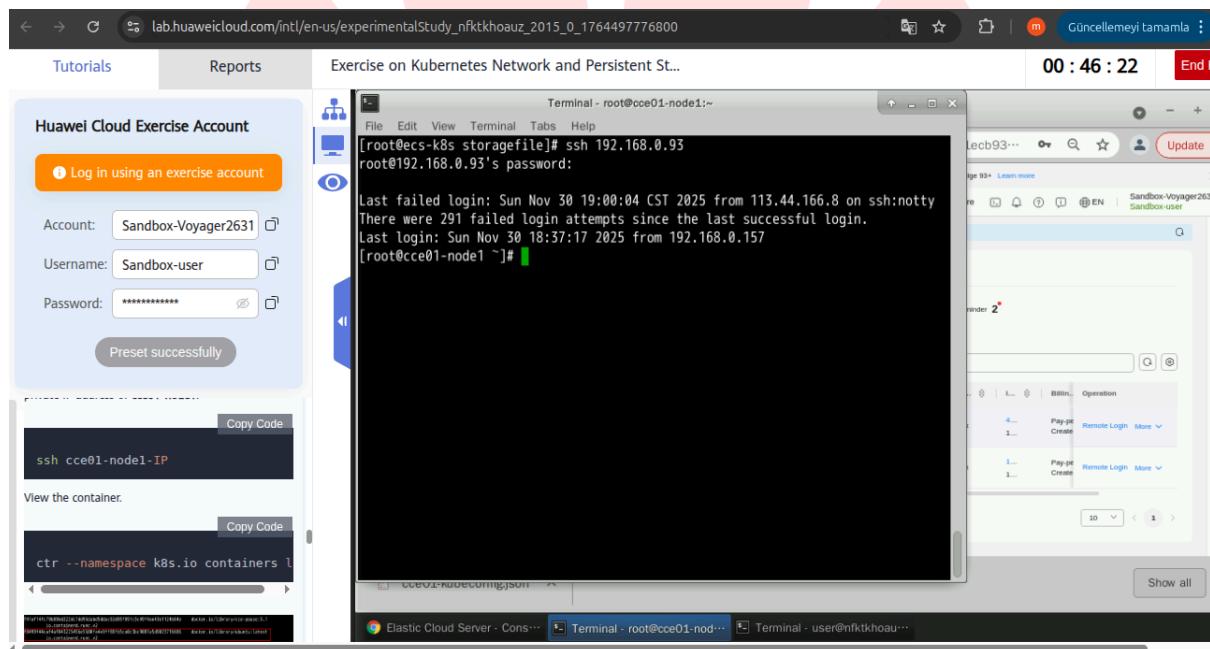
(Görsel 3.4: Pod'un çalışma durumu kontrolü.)

Bu aşamada, oluşturulan **clientpod** içeresine **kubectl exec -it clientpod -- /bin/sh** komutu ile erişilmeye çalışılmış ancak “**container not found**” hatası alınmıştır. Bu durum, pod içerisinde çalıştırılan container adı ve çalışma dizini bilgisinin doğrudan doğru şekilde belirlenmemesinden kaynaklanmıştır.

Daha sonra pod'un çalıştığı node'a (**cce01-node1**) SSH ile bağlanılmak istenmiştir. Ancak denemeler sırasında **Permission denied** hatası alınmıştır. Sorunun, node üzerindeki **root kullanıcı kimlik doğrulaması** ile ilgili olduğu tespit edilmiştir. Çözüm olarak root şifresi yeniden ayarlanmış ve bu işlemden sonra SSH bağlantısı başarıyla gerçekleştirilmiştir.



(Görsel 3.5: Node SSH bağlantı hatası.)



(Görsel 3.6: Hatanın çözümü sonrası bağlantı.)

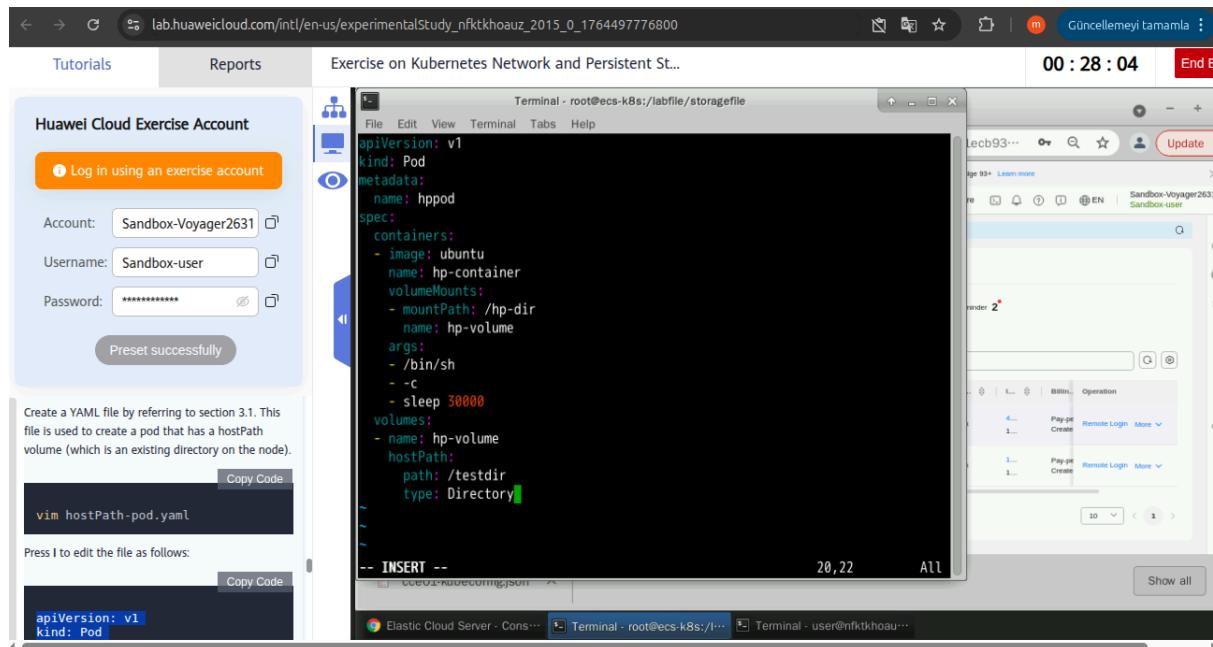
Persistent Storage yapılandırmasında, volume'a ait **ID / mount path bilgisi bulunamadığı** için bağlantı kurulamadı ve ilgili adımda hata alındı. İlk denemelerde beklenen dizin yolu altında gerekli path bulunmadı. Sorunun, volume'un farklı bir dizin altında oluşturulmasından kaynaklandığı anlaşıldı.

Çözüm olarak sistem üzerinde **kök dizinden (/)** başlayarak manuel dizin taraması yapıldı ve volume'a ait gerçek mount path tespit edildi. Doğru path belirlendikten sonra yapılandırma bu dizin üzerinden güncellenmiş ve bağlantı başarılı şekilde sağlanmıştır.

Bu durum lab sürecinde ciddi zaman kaybına neden olmuş ve günün sonunda laboratuvar görevlerinin tamamlanamamasına yol açan iki temel sebepten biri olmuştur.

9) HostPath Volume Kullanılarak Pod Oluşturulması

Bu adımda node üzerindeki **/testdir** dizini hostPath volume olarak tanımlanmış ve pod içerisine mount edilmiştir.



(Görsel 3.7: hostPath volume oluşturma.)

Daha sonra **kubectl apply -f hostpath-pod.yaml** komutu ile uygulanmış daha sonra **kubectl get nodes** komutu ile **hppod** pod'u kontrol edilmiştir.

The screenshot shows the Huawei Cloud Exercise Account interface. On the left, there's a form for logging in with account, username, and password fields, and a "Preset successfully" button. Below it, two terminal windows show the command execution process:

```

kubectl apply -f hostPath-pod.yaml
View the created pod.
Copy Code
kubectl get pod

```

```

[root@ecs-k8s storagefile]# kubectl get pod
NAME      READY   STATUS    RESTARTS   AGE
clientpod 1/1     Running   0          15m
em2        0/1     Evicted   1          3m1s
hppod      1/1     Running   0          3s
nginx-86c57bc6b8-4mksk 1/1     Running   2 (19m ago) 43m
nginx-86c57bc6b8-gkc9h 1/1     Running   2 (19m ago) 43m
nginx-86c57bc6b8-q4v8f  1/1     Running   2 (19m ago) 43m

```

On the right, a terminal window shows the hostPath configuration and pod creation:

```

Terminal - root@ecs-k8s:/labfile/storagefile
File Edit View Terminal Tabs Help
nginx-86c57bc6b8-gkc9h 1/1 Running 2 (17m ago) 41m
nginx-86c57bc6b8-q4v8f 1/1 Running 2 (17m ago) 41m
[root@ecs-k8s storagefile]# ssh 192.168.0.93
-bash: ssh: command not found
[root@ecs-k8s storagefile]# ssh 192.168.0.93
root@192.168.0.93's password:
Last login: Sun Nov 30 19:15:09 2025 from 192.168.0.157
[root@cce01-node1 ~]# mkdir /testdir
[root@cce01-node1 ~]# exit
logout
Connection to 192.168.0.93 closed.
[root@ecs-k8s storagefile]# vim hostPath-pod.yaml
[root@ecs-k8s storagefile]# kubectl apply -f hostPath-pod.yaml
pod/hppod created
[root@ecs-k8s storagefile]# kubectl get pod
NAME      READY   STATUS    RESTARTS   AGE
clientpod 1/1     Running   0          19m ago) 38m
em2        0/1     ContainerStatusUnknown 1          3m1s
hppod      1/1     Running   0          3s
nginx-86c57bc6b8-4mksk 1/1     Running   2 (19m ago) 43m
nginx-86c57bc6b8-gkc9h 1/1     Running   2 (19m ago) 43m
nginx-86c57bc6b8-q4v8f  1/1     Running   2 (19m ago) 43m

```

(Görsel 3.8 : hostPath uygulanması ve kontrolü.)

Pod içerisinde erişim sağlamak amacıyla **kubectl exec** komutu kullanılmıştır. İlgili pod içerisinde girildikten sonra volume'un mount edildiği dizine geçilmiştir. Bu dizin altında test amaçlı bir dosya oluşturulmuş ve içerisinde veri yazılmıştır. Yapılan bu işlem ile hostPath volume'un pod içerisinde doğru şekilde çalıştığı doğrulanmıştır. İşlemler tamamlandıktan sonra pod içerisinde çıkış yapılmıştır.

The screenshot shows the Huawei Cloud Exercise Account interface. On the left, there's a form for logging in with account, username, and password fields, and a "Preset successfully" button. Below it, two terminal windows show the command execution process:

```

EOF
Copy Code
exit

```

Information similar to the following is displayed.

```

[root@ecs-k8s storagefile]# sudo cat >hello2 <<EOF
> hello
> EOF
> exit

```

On the right, a terminal window shows the test file creation and verification:

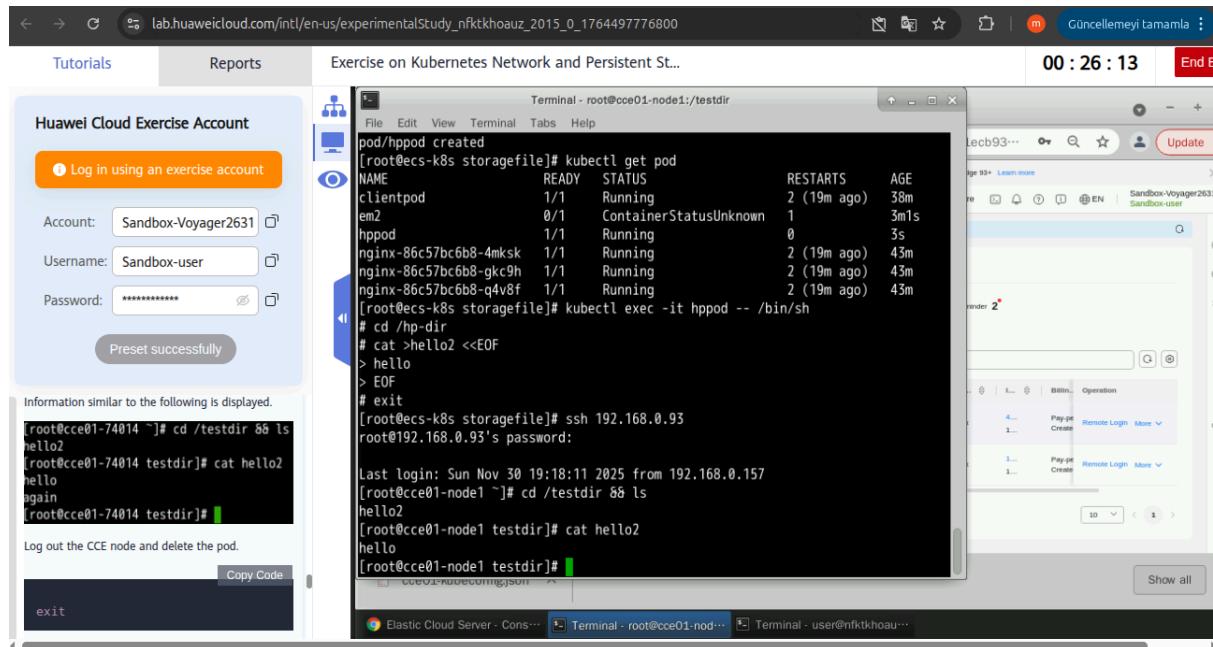
```

[root@ecs-k8s storagefile]# cd /tmp/
[root@ecs-k8s storagefile]# cat >hello2 <<EOF
> hello
> EOF
[root@ecs-k8s storagefile]# cat hello2
hello
[root@ecs-k8s storagefile]# rm hello2

```

(Görsel 3.9: Test amaçlı dosya ve veri yazımı.)

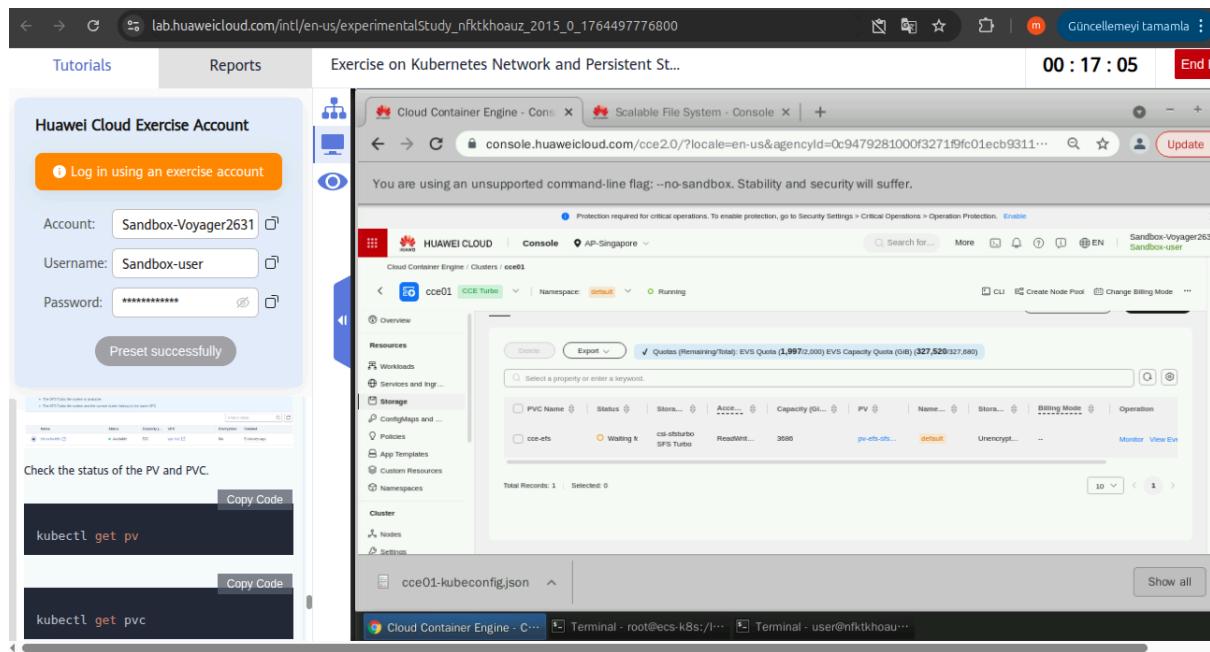
Pod içerisindeki işlemin node tarafına yansıyıp yansımadığını doğrulamak amacıyla pod'dan çıkış yapılmış ve ilgili worker node'a SSH bağlantısı sağlanmıştır. Node üzerinde hostPath volume'un bağlı olduğu dizine gidilerek içerik kontrol edilmiştir. Pod içerisinde oluşturulan dosyanın node tarafında da mevcut olduğu ve içeriğinin aynı şekilde okunabildiği görülmüştür. Bu doğrulama ile pod ve node arasında hostPath volume üzerinden veri paylaşımının başarılı şekilde gerçekleştiği teyit edilmiştir.



(Görsel 4.0 : Pod içerisindeki işlemin node tarafından kontrolü.)

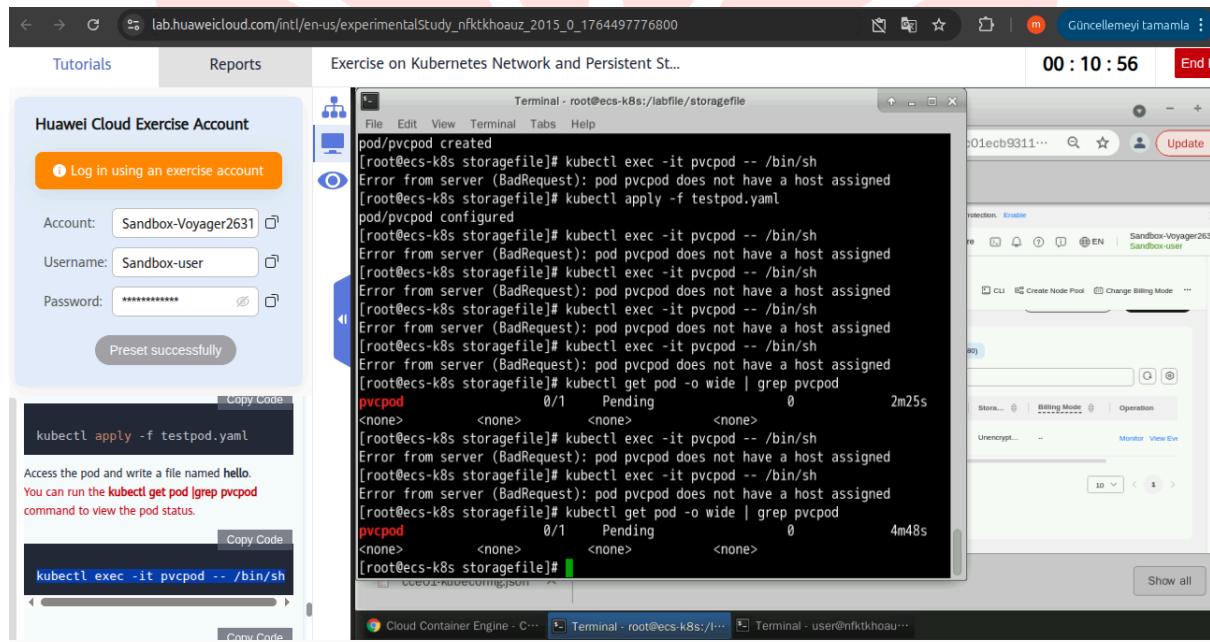
10) PV PVC

Pod için PVC tanımlaması yapıldıktan sonra, gerekli Persistent Volume ve Persistent Volume Claim işlemleri **console üzerinden** tamamlanmıştır.



(Görsel 4.1: PV PVC console üzerinden tanımlanması.)

Ancak PVC'ye bağlı olarak oluşturulan pod, uzun süre **Pending** durumunda kalmış ve herhangi bir node üzerine schedule edilememiştir. Yaklaşık **15 dakika boyunca Pending durumunda beklemesi** nedeniyle pod içerisinde erişim sağlanamamış, dolayısıyla volume doğrulama adımları tamamlanamamıştır. Bu gecikme sebebiyle lab süresi yetersiz kalmış ve labın kalan görevleri tamamlanmadan çalışma sonlandırılmıştır.



(Görsel 4.2: PVC Pending durumunda askıda kalması.)

