

Kubernetes

Day 4

Lab 1

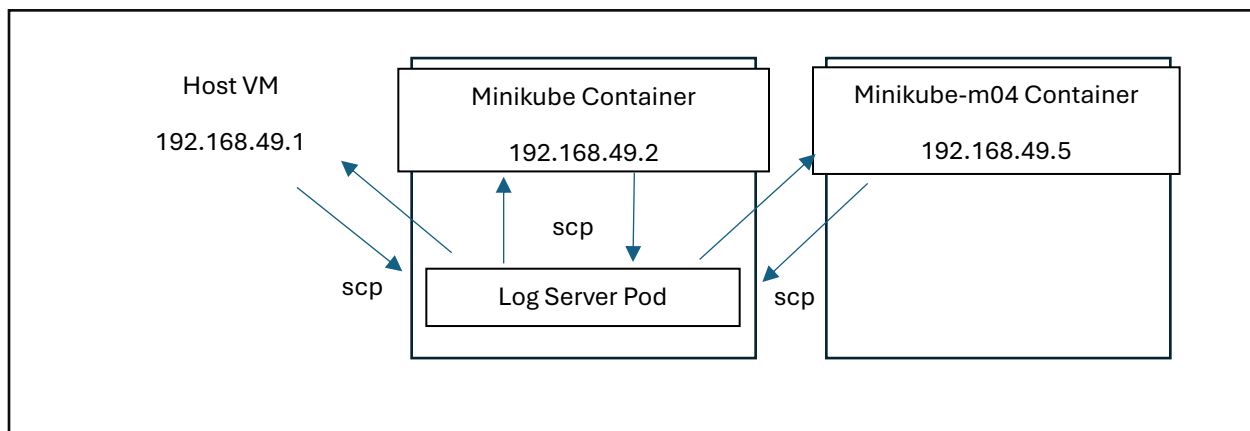
Scheduled Jobs/Cronjobs

Required:

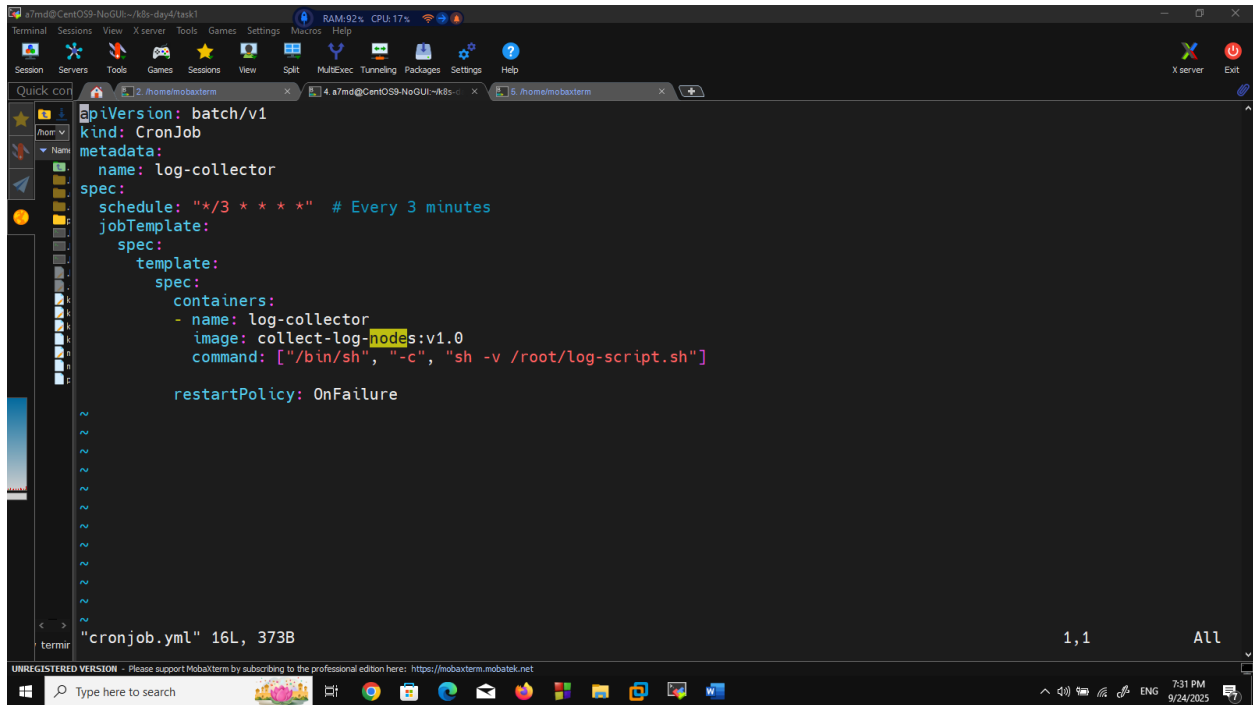
Schedule a Job/Cronjob the reads log files from a cluster's nodes and sends them to the host machine.

Solution:

Cluster's Infrastuctue:



- 1- Created a cronjob .yaml file that will be used to run a periodic job called "log-collector" for reading/collecting jobs every 3 minutes (short interval for testing purposes):



```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: log-collector
spec:
  schedule: "*/3 * * * *" # Every 3 minutes
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: log-collector
              image: collect-log-nodes:v1.0
              command: ["/bin/sh", "-c", "sh -v /root/log-script.sh"]
          restartPolicy: OnFailure
```

"cronjob.yml" 16L, 373B

1,1 All

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

2- The “collect-log-nodes:v1.0” is a basic alpine image built by committing a running container that had openssh installed on it and is able to communicate with the nodes via ssh/scp in a password-less way by a generated shared public key.

This image is created by docker on the main minikube running container. It acts the log-collection server.

```
docker@minikube: ~  
[a7md@CentOS9-NoGUI task1]$ minikube ssh  
docker@minikube:~$  
docker@minikube:~$ docker images | grep alpine  
alpine latest 9234e8fb04c4 2 months ago 8.31MB  
docker@minikube:~$  
docker@minikube:~$ docker images | grep log  
collect-log-nodes v1.0 8fe16d1514d9 7 hours ago 18.1MB  
docker@minikube:~$
```

Now, how to get the nodes IP addresses?

```
a7md@CentOS9-NoGUI ~  
[a7md@CentOS9-NoGUI ~]$ kubectl get nodes -o wide  
NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION  
CONTAINER-RUNTIME  
minikube Ready control-plane 3d1h v1.34.0 192.168.49.2 <none> Ubuntu 22.04.5 LTS 5.14.0-601.e19.x86_6  
4 docker://28.4.0  
[a7md@CentOS9-NoGUI ~]$  
[a7md@CentOS9-NoGUI ~]$ ip addr | grep 192.168.49  
inet 192.168.49.1/24 brd 192.168.49.255 scope global br-3fe243b155c7  
[a7md@CentOS9-NoGUI ~]$  
[a7md@CentOS9-NoGUI ~]$
```

The “minikube-m04” was deleted before this document was written, but we still can see it’s still running via “minikube status” command on the host.

```
[a7md@CentOS9-NoGUI ~]$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

minikube-m02
type: Worker
host: Running
kubelet: Running

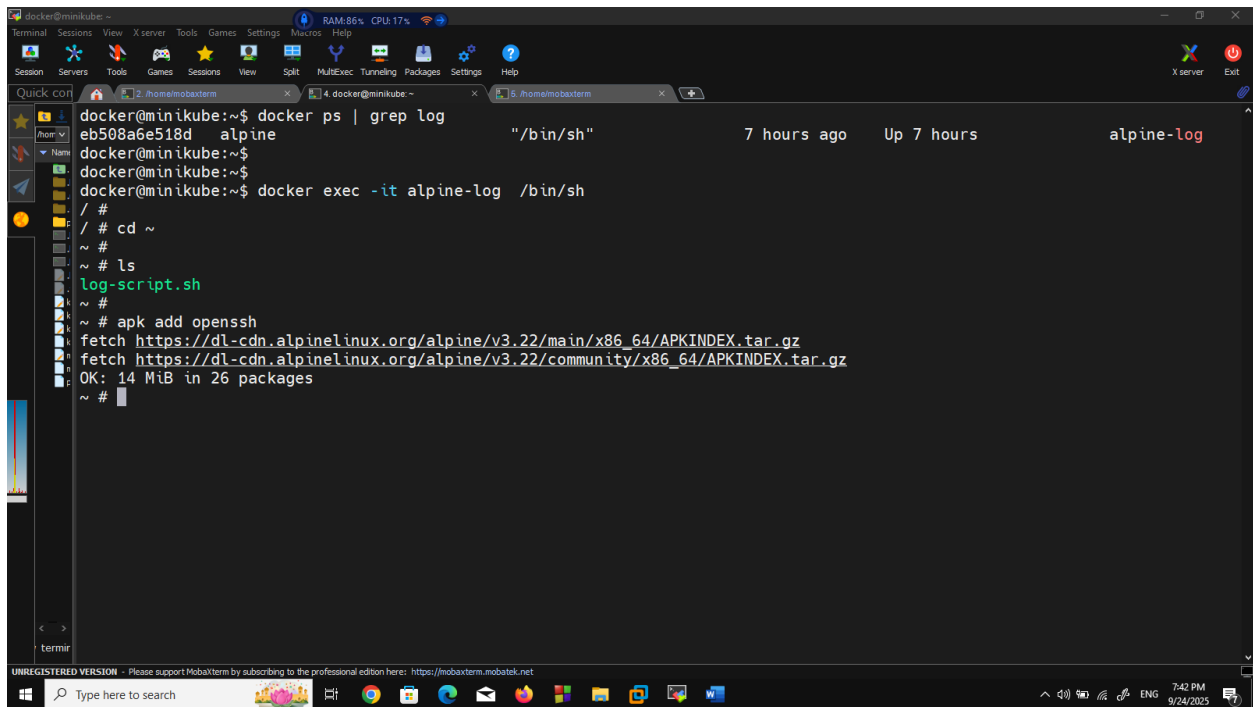
minikube-m03
type: Worker
host: Running
kubelet: Stopped

minikube-m04
type: Worker
host: Running
kubelet: Running

[a7md@CentOS9-NoGUI ~]$
```

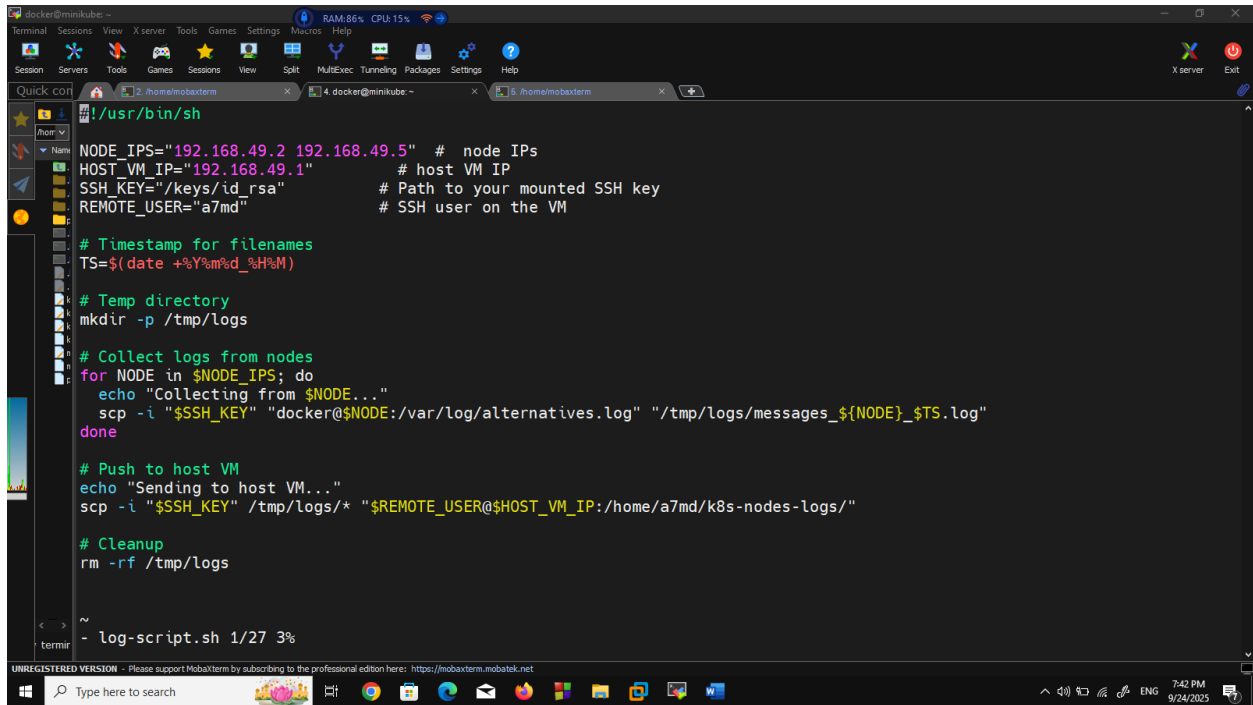
```
[a7md@CentOS9-NoGUI ~]$ minikube ssh --node minikube-m04
docker@minikube-m04:~$
docker@minikube-m04:~$
docker@minikube-m04:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0@if14: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether f2:21:63:4a:0d:95 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.49.5/24 brd 192.168.49.255 scope global eth0
        valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether e2:97:61:41:d2:70 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
4: flannel.1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default
    link/ether a6:d4:2e:31:6e:df brd ff:ff:ff:ff:ff:ff
    inet 10.244.1.0/32 scope global flannel.1
        valid_lft forever preferred_lft forever
    inet6 fe80::a4d4:2eff:fe31:6edf/64 scope link
        valid_lft forever preferred_lft forever
5: cni0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 5e:d8:83:f5:a9:cf brd ff:ff:ff:ff:ff:ff
    inet 10.244.1.1/24 brd 10.244.1.255 scope global cni0
        valid_lft forever preferred_lft forever
    inet6 fe80::5cd8:83ff:fef5:a9cf/64 scope link
```

3- Let's see what was added on the container to be committed to a new image



```
docker@minikube:~$ docker ps | grep log
eb508a6e518d  alpine          "/bin/sh"          7 hours ago    Up 7 hours        alpine-log
docker@minikube:~$
docker@minikube:~$ docker exec -it alpine-log /bin/sh
/ #
/ # cd ~
~ #
~ # ls
log-script.sh
~ #
~ # apk add openssh
fetch https://dl-cdn.alpinelinux.org/alpine/v3.22/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.22/community/x86_64/APKINDEX.tar.gz
OK: 14 MiB in 26 packages
~ #
```

Figure 1: Installing OpenSSH

A screenshot of a MobaXterm terminal window. The terminal is running a shell prompt `docker@mimikube: ~`. The user has entered `!usr/bin/sh`. The script content is as follows:

```
NODE_IPS="192.168.49.2 192.168.49.5" # node IPs
HOST_VM_IP="192.168.49.1" # host VM IP
SSH_KEY="/keys/id_rsa" # Path to your mounted SSH key
REMOTE_USER="a7md" # SSH user on the VM

# Timestamp for filenames
TS=$(date +%Y%m%d_%H%M)

# Temp directory
mkdir -p /tmp/logs

# Collect logs from nodes
for NODE in $NODE_IPS; do
    echo "Collecting from $NODE..."
    scp -i "$SSH_KEY" "docker@$NODE:/var/log/alternatives.log" "/tmp/logs/messages_${NODE}_${TS}.log"
done

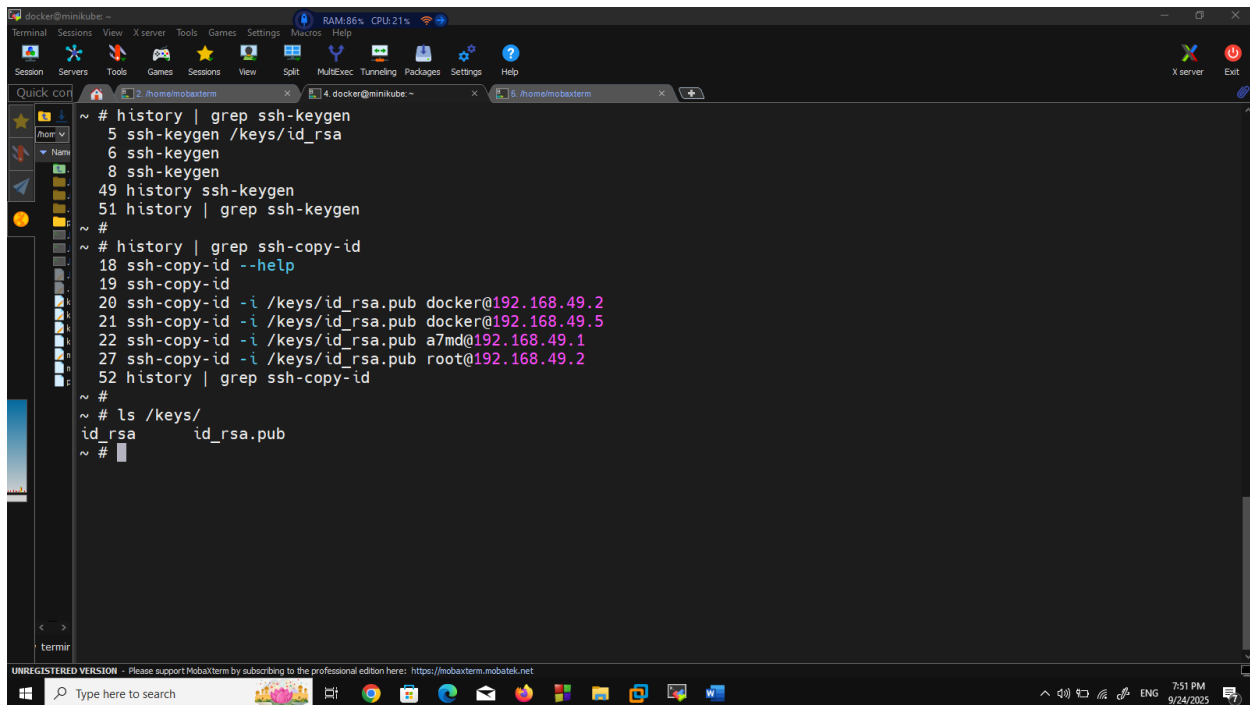
# Push to host VM
echo "Sending to host VM..."
scp -i "$SSH_KEY" /tmp/logs/* "$REMOTE_USER@$HOST_VM_IP:/home/a7md/k8s-nodes-logs/"

# Cleanup
rm -rf /tmp/logs
```

The terminal shows the script is being executed, with a progress bar at the bottom indicating `- log-script.sh 1/27 3%`. The MobaXterm interface includes a menu bar, a toolbar, and a sidebar with file explorer and session manager views. The Windows taskbar is visible at the bottom.

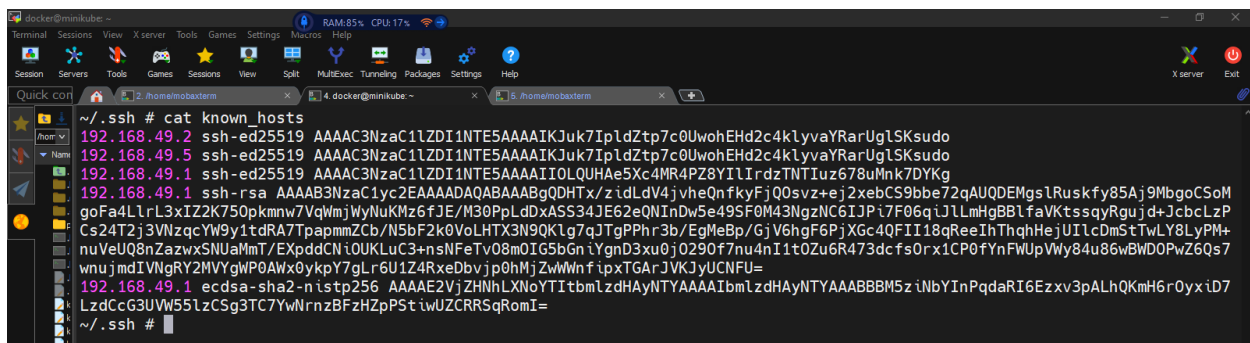
Figure 2: Adding log-collection script

The log-collection script copies a certain file with read permissions on the node containers, to a file uniquely named after the node IP and time stamp it was copied at.



```
docker@minikube: ~  
# history | grep ssh-keygen  
5 ssh-keygen /keys/id_rsa  
6 ssh-keygen  
8 ssh-keygen  
49 history ssh-keygen  
51 history | grep ssh-keygen  
#  
# history | grep ssh-copy-id  
18 ssh-copy-id --help  
19 ssh-copy-id  
20 ssh-copy-id -i /keys/id_rsa.pub docker@192.168.49.2  
21 ssh-copy-id -i /keys/id_rsa.pub docker@192.168.49.5  
22 ssh-copy-id -i /keys/id_rsa.pub a7md@192.168.49.1  
27 ssh-copy-id -i /keys/id_rsa.pub root@192.168.49.2  
52 history | grep ssh-copy-id  
#  
# ls /keys/  
id_rsa      id_rsa.pub  
#
```

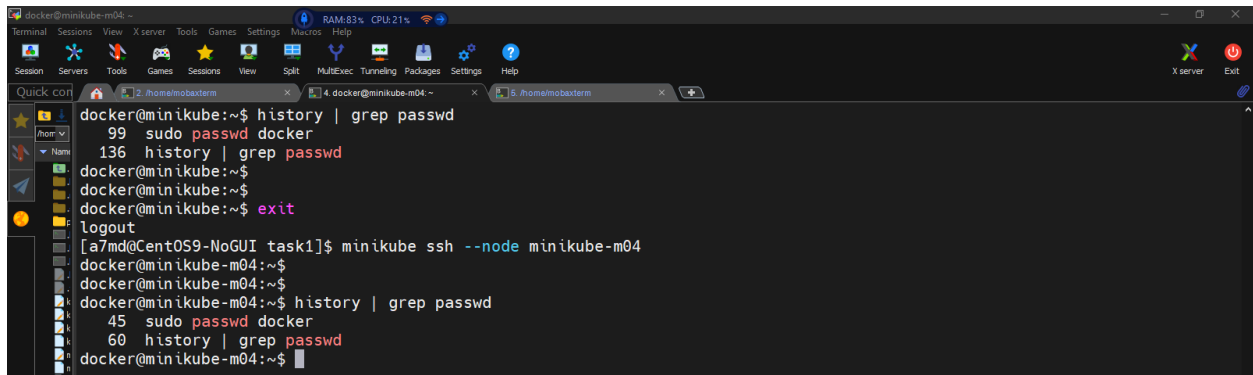
Figure 3: Copy a generated RSA Public key for SCP protocol



```
docker@minikube: ~  
# cat known_hosts  
192.168.49.2 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKJuk7IpLdZtp7c0UwohEHd2c4klyvaYRarUgLSKsdo  
192.168.49.5 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKJuk7IpLdZtp7c0UwohEHd2c4klyvaYRarUgLSKsdo  
192.168.49.1 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOLQUHAe5Xc4MR4PZ8YI1IrdzTNTIuz678uMnk7DYKq  
192.168.49.1 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDHTx/zidLdV4jvheQnfkyFjQ0svz+ej2xabCS9bbe72qAUQDEMGs1Ruskfy85Aj9MbgoCSom  
goFa4LrL3xIZ2K750pkmnw7VqWmJWynNukMz6fJE/M30PpLdDxASS34JE62eQNIInDw5e49SF0M43NgzNC6IJPi7F06qiJlLmHgBBLfVKTssqyRgujd+JcblZP  
Cs24T2j3VNzqcYw9y1tdRA7TpapmmZCb/N5bF2k0VoLHTX3N9QKlg7qJTgPPHr3b/EgMeBp/GjV6hgF6PjXGc4QFI18qReeIhThqhHejUIlcDmStTwLY8LyPM+  
nuVeUQ8nZazwxSNUaMmT/EXpddCNiOUKLuC3+nNFeTv08m0IG5bGn iYgnD3xu0j0290f7nu4nI1t0Zu6R473dcfs0rx1CP0fYnFWUpVwy84u86wBWD0PwZ6Qs7  
wnuJmDIVNgRY2MVYgWP0AWx0ykpY7gLR6U1Z4RxeDbvjp0hMjZwWwnfipxTGAJVKJyUCNFU=  
192.168.49.1 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlkbmdhYXNTYAAAABmLzdhYXNTYAAABBM5ziNbYInPqdaRI6Ezxv3pALhQKmH6rOyxID7  
LzdCcG3UWV55lZCSg3TC7YwNrnzBFzHZpPStiwUZCRRSQRomI=  
#
```

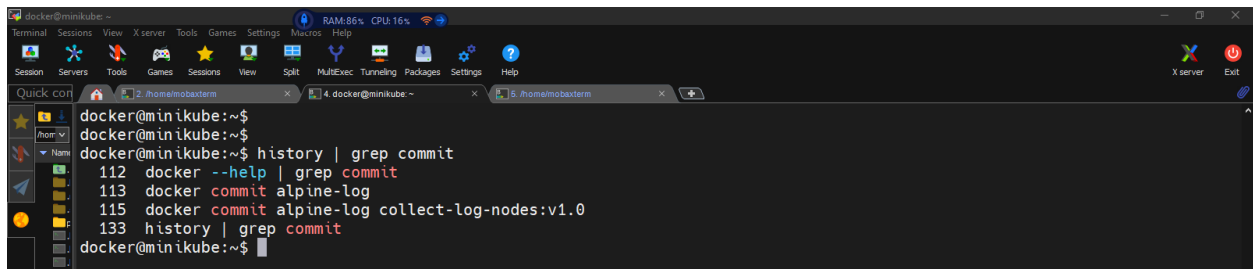
This is a mandatory step before running the script in order to get it working in a password-less way.

This operation required the passwords of the “docker” users on the two nodes “minikube” and “minikube-m04”, which was changed via “sudo passwd docker” command, which seemed like that they were “sudoer” users, hopefully.

A terminal window titled 'docker@minikube-m04: ~' showing a sequence of commands to change the password of the 'docker' user. The user runs 'history | grep passwd' showing previous attempts. Then 'sudo passwd docker' is executed. The user logs out and SSHs into the 'minikube-m04' node. Inside the node, 'history | grep passwd' shows the previous command, and 'sudo passwd docker' is run again to confirm the password change.

```
docker@minikube-m04: ~$ history | grep passwd
99  sudo passwd docker
136 history | grep passwd
docker@minikube-m04: ~$ passwd
docker@minikube-m04: ~$ exit
logout
[a7md@CentOS9-NoGUI task1]$ minikube ssh --node minikube-m04
docker@minikube-m04: ~$
docker@minikube-m04: ~$ history | grep passwd
45  sudo passwd docker
60  history | grep passwd
docker@minikube-m04: ~$
```

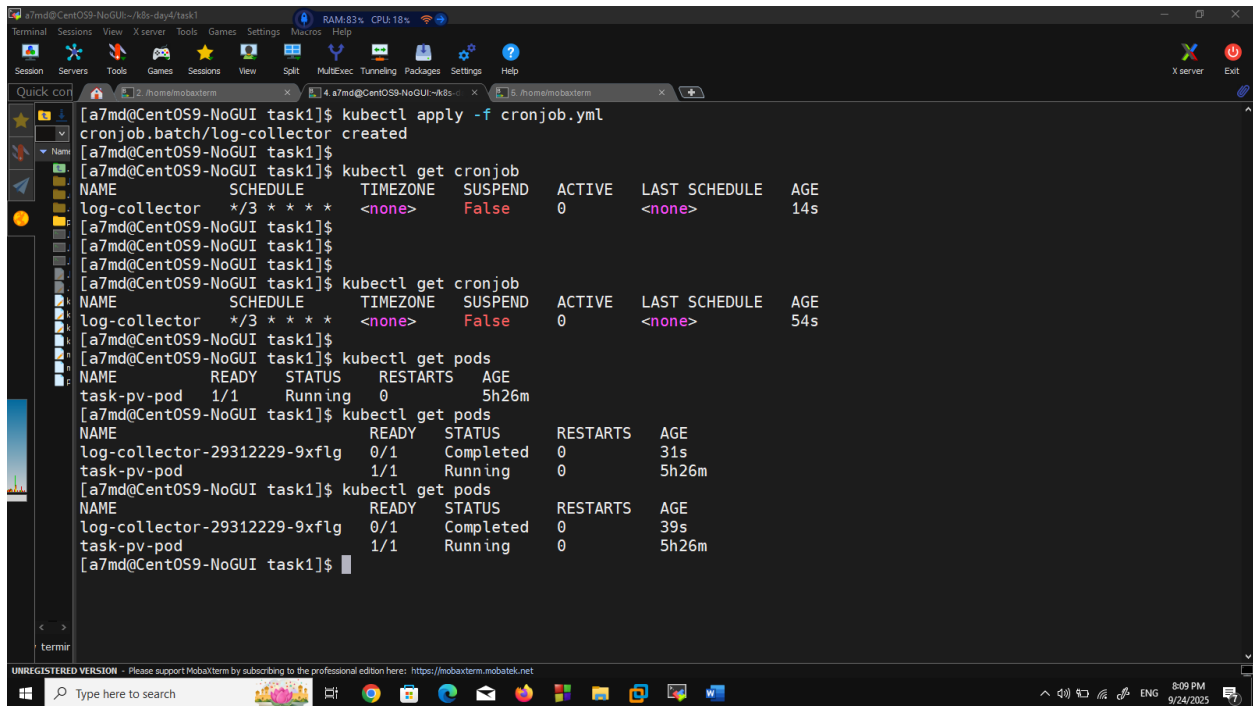
Figure 4: Changing the 'docker' user's password on both the minikube container

A terminal window titled 'docker@minikube: ~' showing the user running 'history | grep commit' to find the commit command. The history shows a previous attempt with '--help' and a successful commit of the 'alpine-log' container. The user then runs 'docker commit alpine-log collect-log-nodes:v1.0' to save the current state of the 'alpine-log' container as a new image.

```
docker@minikube: ~$
docker@minikube: ~$
docker@minikube: ~$ history | grep commit
112 docker --help | grep commit
113 docker commit alpine-log
115 docker commit alpine-log collect-log-nodes:v1.0
133 history | grep commit
docker@minikube: ~$
```

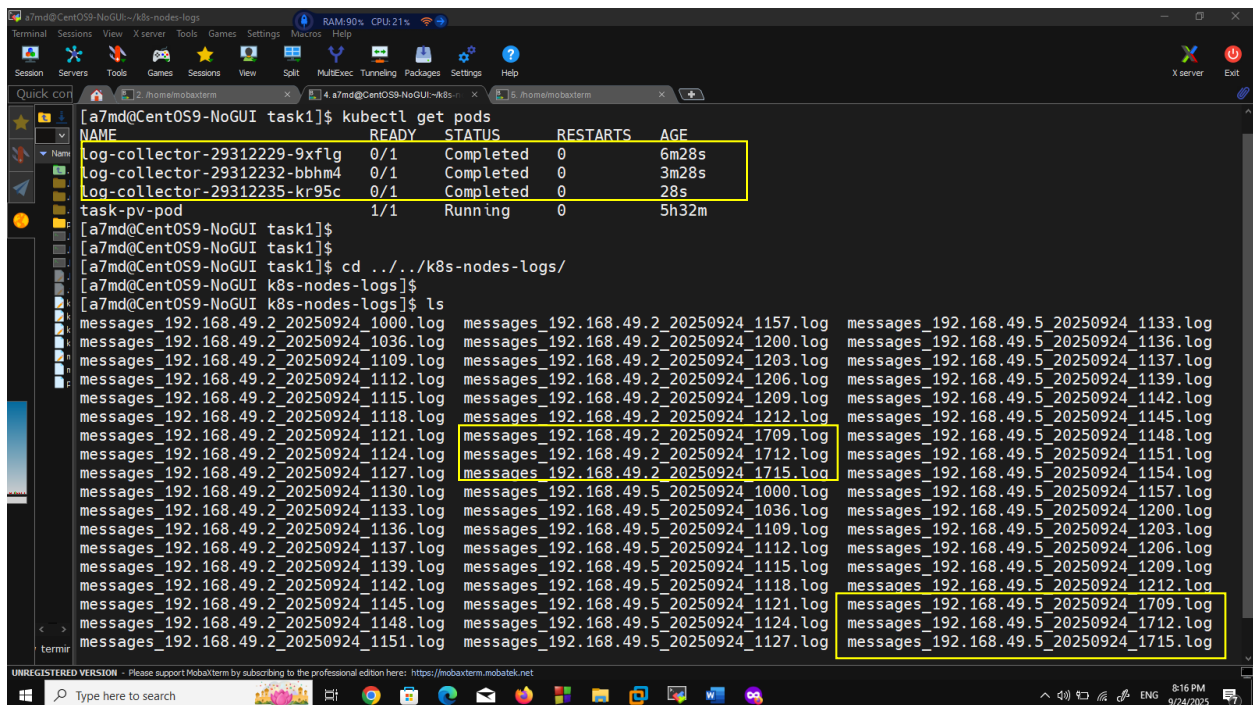
Figure 5: Committing the container at this state to an image used by the pod created by the cronjob

4- Starting the cronjob:



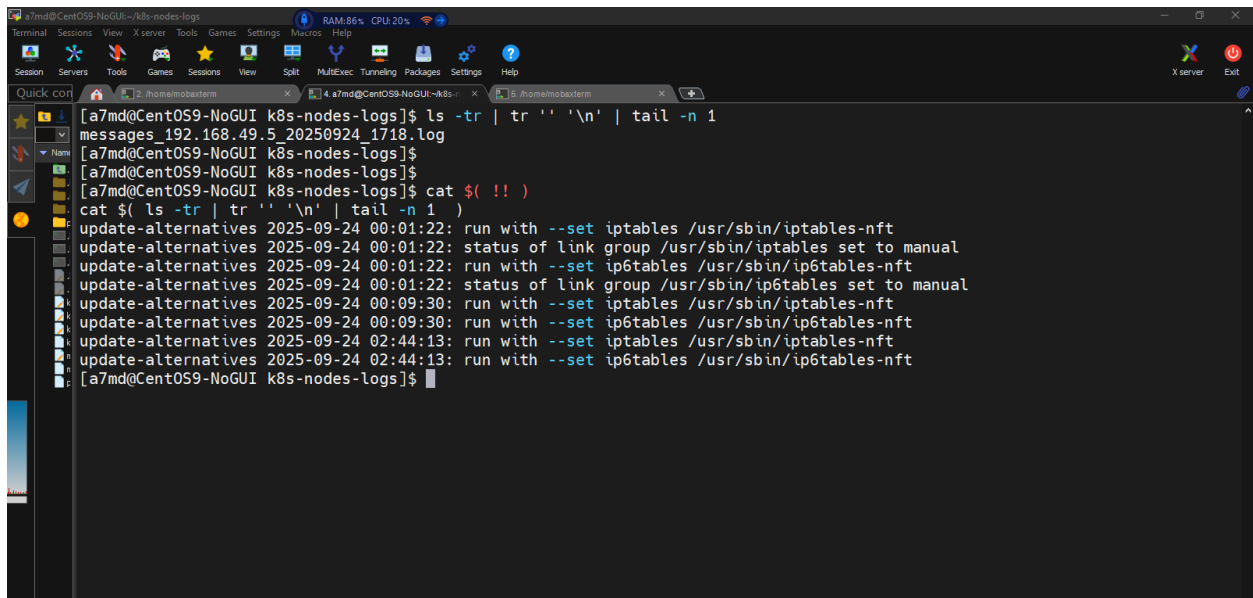
```
[a7md@CentOS9-NoGUI task1]$ kubectl apply -f cronjob.yml
cronjob.batch/log-collector created
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ kubectl get cronjob
NAME          SCHEDULE    TIMEZONE   SUSPEND    ACTIVE   LAST SCHEDULE   AGE
log-collector */3 * * * *    <none>     False      0          <none>    14s
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ kubectl get cronjob
NAME          SCHEDULE    TIMEZONE   SUSPEND    ACTIVE   LAST SCHEDULE   AGE
log-collector */3 * * * *    <none>     False      0          <none>    54s
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
task-pv-pod   1/1     Running   0           5h26m
[a7md@CentOS9-NoGUI task1]$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
log-collector-29312229-9xflg  0/1     Completed  0           31s
task-pv-pod   1/1     Running   0           5h26m
[a7md@CentOS9-NoGUI task1]$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
log-collector-29312229-9xflg  0/1     Completed  0           39s
task-pv-pod   1/1     Running   0           5h26m
[a7md@CentOS9-NoGUI task1]$
```

After some time:



```
[a7md@CentOS9-NoGUI task1]$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
log-collector-29312229-9xflg  0/1     Completed  0           6m28s
log-collector-29312232-bbhm4  0/1     Completed  0           3m28s
log-collector-29312235-kr95c  0/1     Completed  0           28s
task-pv-pod   1/1     Running   0           5h32m
[a7md@CentOS9-NoGUI task1]$
[a7md@CentOS9-NoGUI task1]$ cd ../../k8s-nodes-logs/
[a7md@CentOS9-NoGUI k8s-nodes-logs]$ ls
messages_192.168.49.2_20250924_1000.log messages_192.168.49.2_20250924_1000.log messages_192.168.49.5_20250924_1133.log
messages_192.168.49.2_20250924_1036.log messages_192.168.49.2_20250924_1200.log messages_192.168.49.5_20250924_1136.log
messages_192.168.49.2_20250924_1109.log messages_192.168.49.2_20250924_1203.log messages_192.168.49.5_20250924_1137.log
messages_192.168.49.2_20250924_1112.log messages_192.168.49.2_20250924_1206.log messages_192.168.49.5_20250924_1139.log
messages_192.168.49.2_20250924_1115.log messages_192.168.49.2_20250924_1209.log messages_192.168.49.5_20250924_1142.log
messages_192.168.49.2_20250924_1118.log messages_192.168.49.2_20250924_1212.log messages_192.168.49.5_20250924_1145.log
messages_192.168.49.2_20250924_1121.log messages_192.168.49.2_20250924_1709.log messages_192.168.49.5_20250924_1148.log
messages_192.168.49.2_20250924_1124.log messages_192.168.49.2_20250924_1712.log messages_192.168.49.5_20250924_1151.log
messages_192.168.49.2_20250924_1127.log messages_192.168.49.2_20250924_1715.log messages_192.168.49.5_20250924_1154.log
messages_192.168.49.2_20250924_1130.log messages_192.168.49.5_20250924_1000.log messages_192.168.49.5_20250924_1157.log
messages_192.168.49.2_20250924_1133.log messages_192.168.49.5_20250924_1036.log messages_192.168.49.5_20250924_1200.log
messages_192.168.49.2_20250924_1136.log messages_192.168.49.5_20250924_1109.log messages_192.168.49.5_20250924_1203.log
messages_192.168.49.2_20250924_1137.log messages_192.168.49.5_20250924_1112.log messages_192.168.49.5_20250924_1206.log
messages_192.168.49.2_20250924_1139.log messages_192.168.49.5_20250924_1115.log messages_192.168.49.5_20250924_1209.log
messages_192.168.49.2_20250924_1142.log messages_192.168.49.5_20250924_1118.log messages_192.168.49.5_20250924_1212.log
messages_192.168.49.2_20250924_1145.log messages_192.168.49.5_20250924_1121.log messages_192.168.49.5_20250924_1709.log
messages_192.168.49.2_20250924_1148.log messages_192.168.49.5_20250924_1124.log messages_192.168.49.5_20250924_1712.log
messages_192.168.49.2_20250924_1151.log messages_192.168.49.5_20250924_1127.log messages_192.168.49.5_20250924_1715.log
```

We can see the last three recently copied log files from each node, by the 3 pods marked as Completed.

A screenshot of a terminal window titled 'a7md@CentOS9-NoGUI ~/k8s-nodes-logs'. The terminal shows the output of several commands. First, 'ls -tr | tr '\n' | tail -n 1' lists the most recent log file: 'messages_192.168.49.5_20250924_1718.log'. Then, 'cat \$(ls -tr | tr '\n' | tail -n 1)' displays the contents of this log file. The log entries show 'update-alternatives' commands being run on 2025-09-24 at various times, with the status of link groups for 'iptables' and 'ip6tables' being set to 'manual'.

```
[a7md@CentOS9-NoGUI k8s-nodes-logs]$ ls -tr | tr '\n' | tail -n 1
messages_192.168.49.5_20250924_1718.log
[a7md@CentOS9-NoGUI k8s-nodes-logs]$
[a7md@CentOS9-NoGUI k8s-nodes-logs]$ cat $(!!)
cat $(ls -tr | tr '\n' | tail -n 1)
update-alternatives 2025-09-24 00:01:22: run with --set iptables /usr/sbin/iptables-nft
update-alternatives 2025-09-24 00:01:22: status of link group /usr/sbin/iptables set to manual
update-alternatives 2025-09-24 00:01:22: run with --set ip6tables /usr/sbin/ip6tables-nft
update-alternatives 2025-09-24 00:01:22: status of link group /usr/sbin/ip6tables set to manual
update-alternatives 2025-09-24 00:09:30: run with --set iptables /usr/sbin/iptables-nft
update-alternatives 2025-09-24 00:09:30: run with --set ip6tables /usr/sbin/ip6tables-nft
update-alternatives 2025-09-24 02:44:13: run with --set iptables /usr/sbin/iptables-nft
update-alternatives 2025-09-24 02:44:13: run with --set ip6tables /usr/sbin/ip6tables-nft
[a7md@CentOS9-NoGUI k8s-nodes-logs]$
```

Figure 6: Looking at the last-received log file