CISA 4390
Managing Network Services
(Final Project)

Submitted by: **Group 3**

Ayush Gusain

Henry Hua

Jabez Barredo

Jianli Yang

Nelson Tulio

Tyler Ko

Table of Contents

YouTube Link: https://youtu.be/NauJV7uuHKU

Project Explanation

In this project, we installed a 3-node Proxmox VE cluster running on bare-metal systems. With a distributed storage and OSD configured the high availability using Ceph. And installed three virtual machines with Linux Mint OS on each node, and those virtual machines fail over between nodes when the hardware failure occurs.

And on each node/virtual machine we installed a microk8s Kubernetes environment and created a cluster in node1 and configured the join command to Node2 and Node3. And deployed containerized AI services including Ollama, OpenClaw and LobeChat. Containerization is a way of packaging an application with everything it needs to run like the files or libraries, the codes and dependencies. With this, it allows the application to run consistently on different systems without needing a manual setup.

Infrastructure Overview

The infrastructure was designed as a fully self-hosted and highly available platform using free and open-source software.

We use three physical computers with Proxmox VE installed on all of them. And the three computers were connected together as a cluster. We did that so they could work as one system because this project needs a lot of hardware resources to make it work smoothly as intended. And in Proxmox VE we used Ceph to share storage between the machines with high availability configured that allows virtual machines to automatically restart on another node to reduce downtime when the hardware failure occurs.

And inside of the Proxmox cluster, we deploy three virtual machines with Linux mint installed on it. These virtual machines are connected to each other using microK8s. We use microK8s because this is the lightweight version of Kubernetes which is used to manage containers and applications.

Using the Kubernetes cluster, we configured and deployed Ollama so we can run AI models locally, and we added Qwen, DeepSeek, and Llama directly into it so the AI can run completely into our own machine without needing internet AI services. OpenClaw for automation of tasks and Lobe chat as the chatbot with a website interface.

Part 1: Setting up the Physical Environment

1.1 Physical Nodes and Virtual Machine Addressing

Node Name	Node Virtual Machine	IP Address
Group3-Node01		142.232.241.222/24
	100 (Group3-Vm01)	142.232.241.68/24
Group3-Node02		142.232.241.221/24
	101 (Group3-Vm02)	142.232.241.67/24

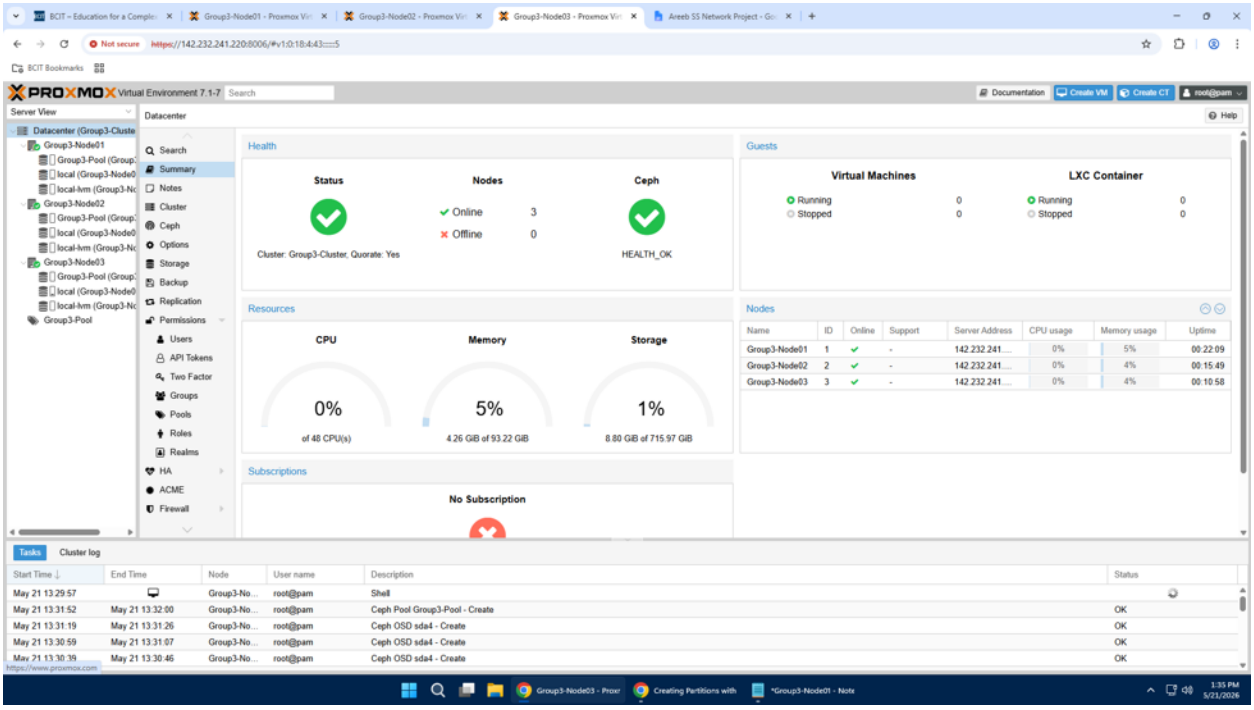
Group3-Node03		142.232.241.220/24
	102 (Group3-Vm03)	142.232.241.65/24

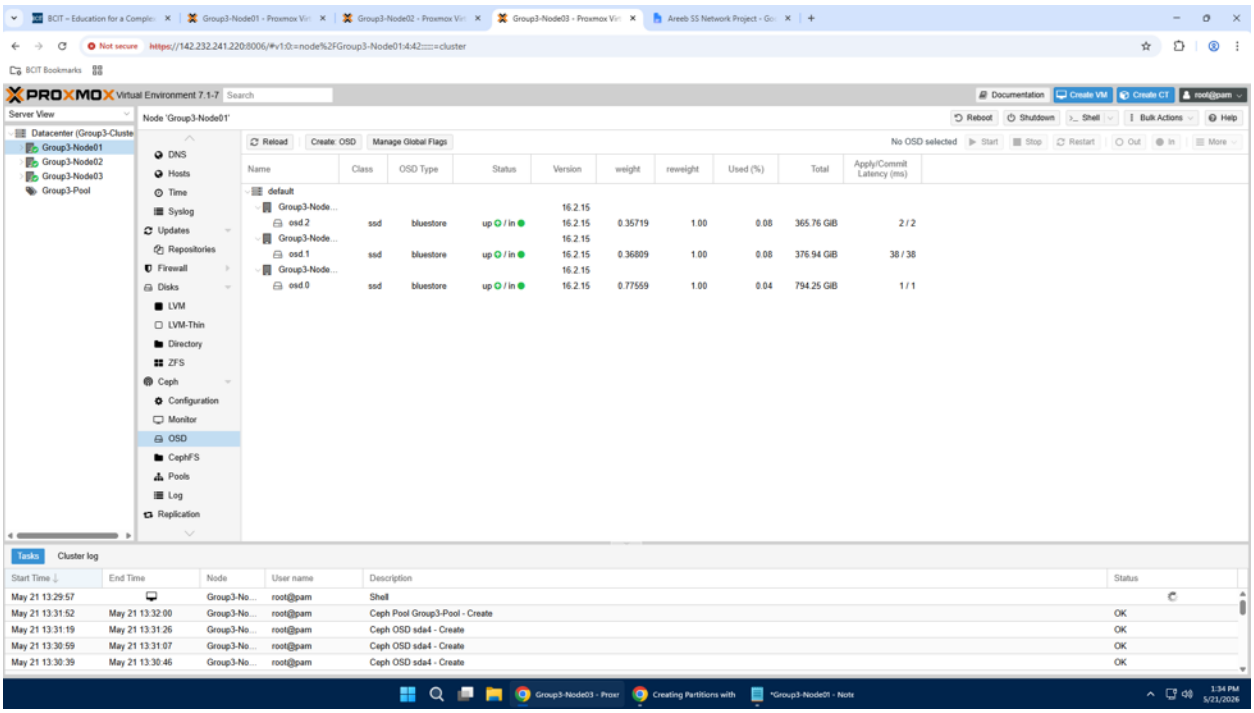
1.2 Installing Proxmox VE

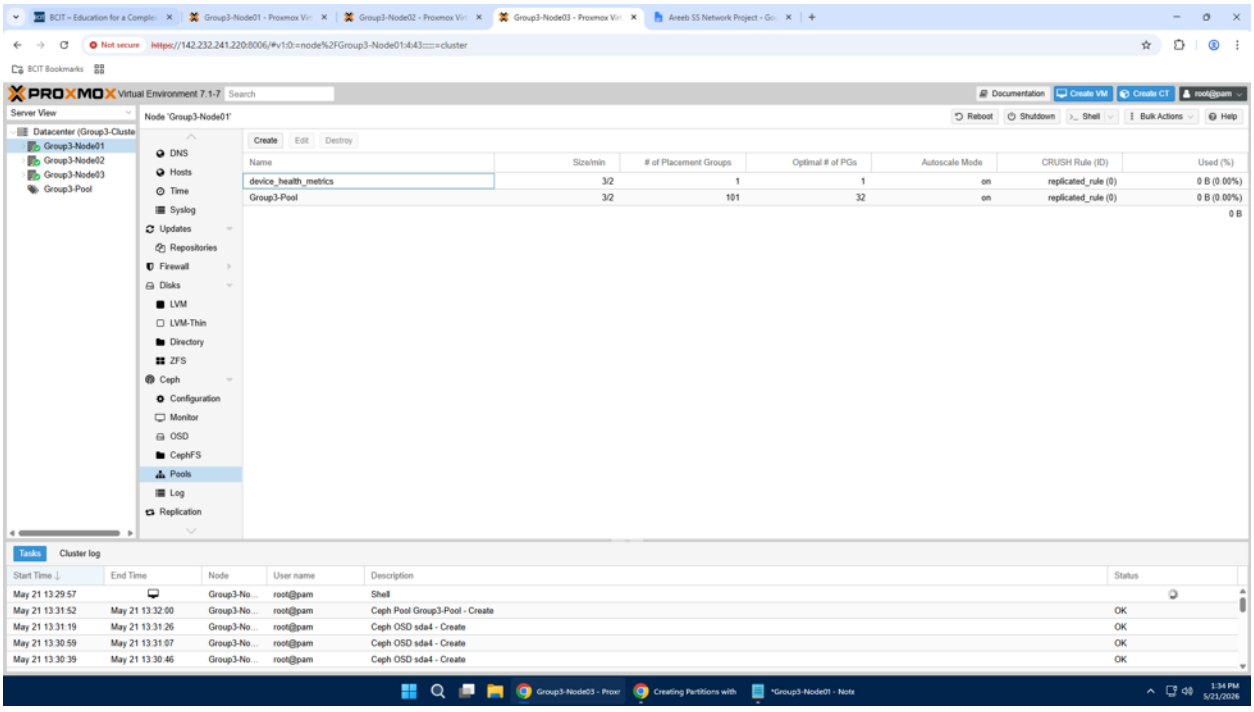
Each machine has a Proxmox VE installed with a static IP address and is able to reach the internet. Only assigned 100GB of storage for Proxmox VE OS and the rest of the unallocated space is used for OSD in Ceph.

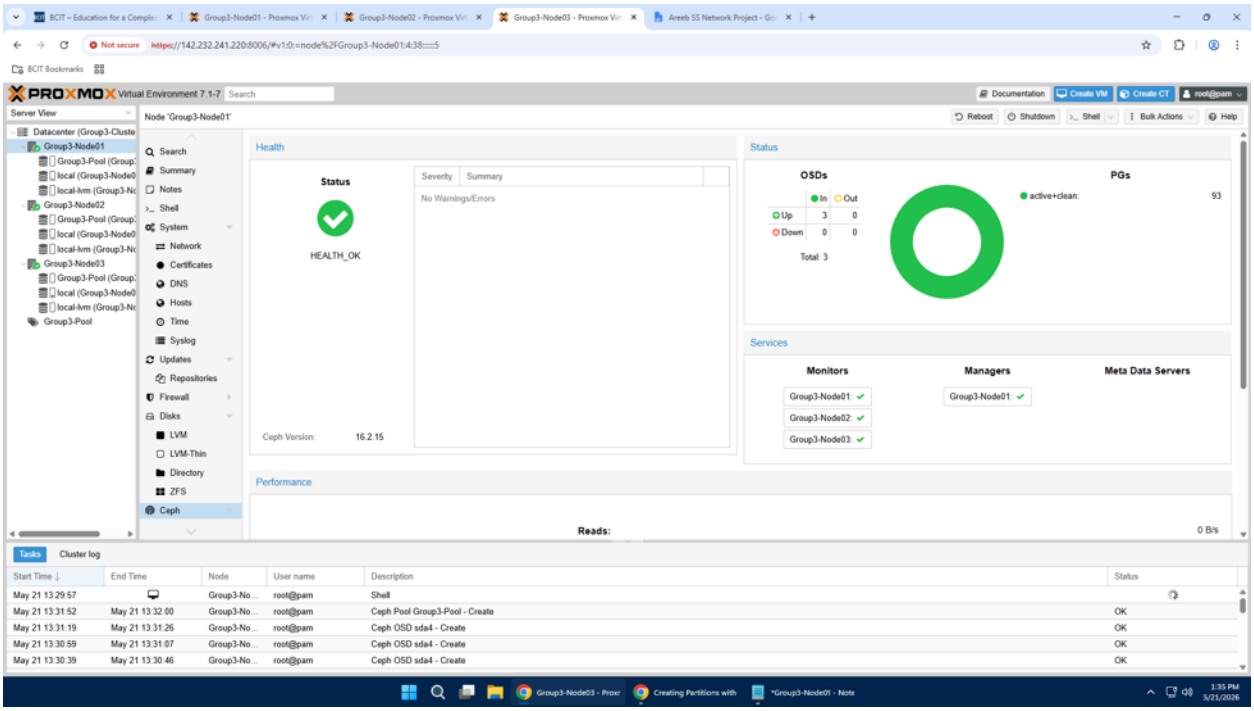
1.3 Install Ceph and Create OSD and a Resource Pool

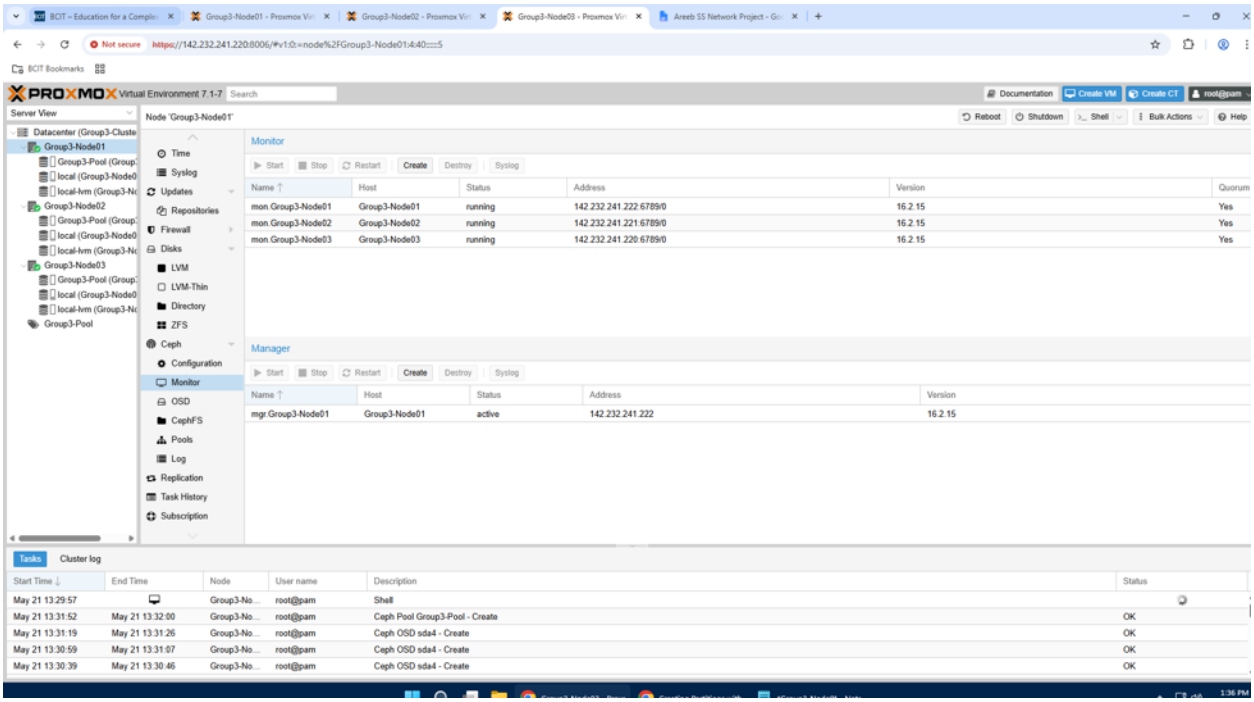
Each node has Ceph installed, and the remaining storage space from each node is used as OSD, and a resource pool named “Group3-Pool” is created for high availability. And assigned monitors and managers for quorum. And also, the High Availability is enabled.




Virtual Environment 7.1-7
Node 'Group3-Node01'
Datacenter (Group3-Cluste
Group3-Node01
Group3-Node02
Group3-Node03
Group3-Pool
DNS
Hosts
Time
Syslog
Updates
Repositories
Firewall
Disks
LVM
LVM-Thin
Directory
ZFS
Ceph
Configuration
Monitor
OSD
CephFS
Pools
Log
Replication
Reload
Create: OSD
Manage Global Flags
No OSD selected
Name
Class
OSD Type
Status
Version
weight
reweight
Used (%)
Total
Apply/Commit Latency (ms)
default
Group3-Node...
osd.2 ssd bluestore up / in 16.2.15 0.35719 1.00 0.08 365.76 GiB 2 / 2
osd.1 ssd bluestore up / in 16.2.15 0.36809 1.00 0.08 376.94 GiB 38 / 38
osd.0 ssd bluestore up / in 16.2.15 0.77559 1.00 0.04 794.25 GiB 1 / 1
Tasks
Cluster log
Start Time
End Time
Node
User name
Description
Status
May 21 13:29:57 Group3-No... root@pam Shell
May 21 13:31:52 May 21 13:32:00 Group3-No... root@pam Ceph Pool Group3-Pool - Create OK
May 21 13:31:19 May 21 13:31:26 Group3-No... root@pam Ceph OSD sda4 - Create OK
May 21 13:30:59 May 21 13:31:07 Group3-No... root@pam Ceph OSD sda4 - Create OK
May 21 13:30:39 May 21 13:30:46 Group3-No... root@pam Ceph OSD sda4 - Create OK
1:34 PM
5/21/2026

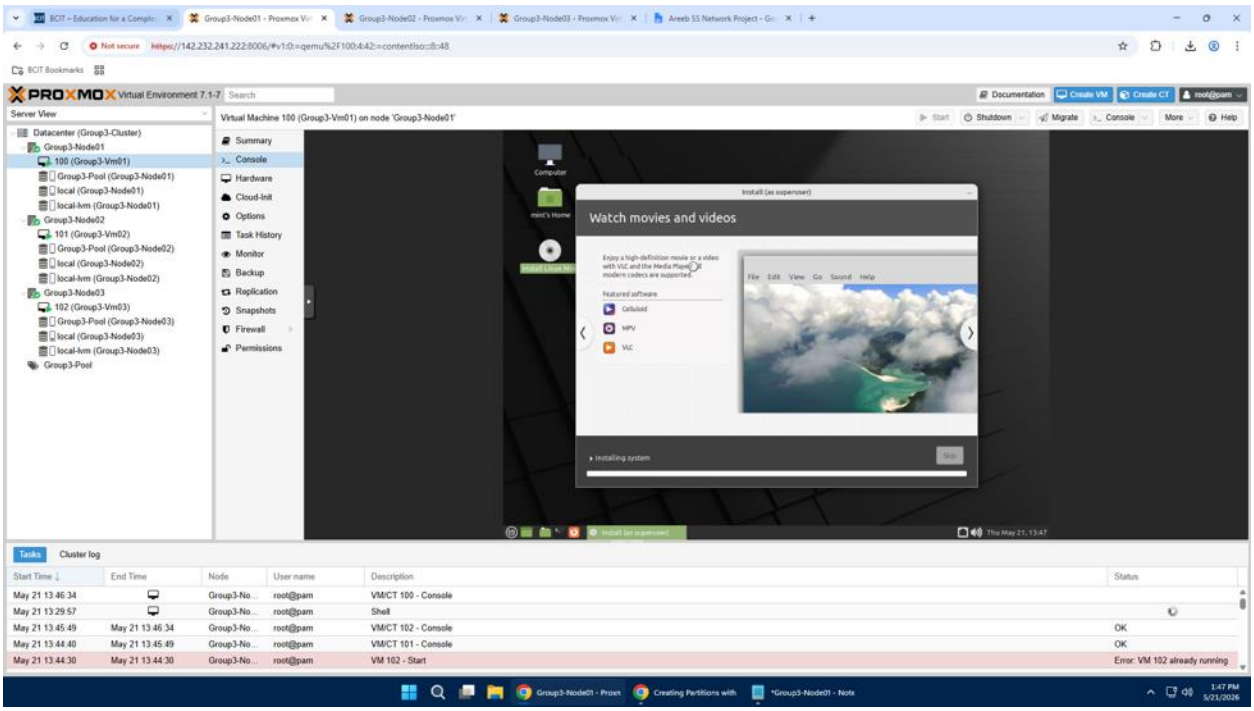

Virtual Environment 7.1-7
Node 'Group3-Node01'
Datacenter (Group3-Cluste
Group3-Node01
Group3-Node02
Group3-Node03
Group3-Pool
DNS
Hosts
Time
Syslog
Updates
Repositories
Firewall
Disks
LVM
LVM-Thin
Directory
ZFS
Ceph
Configuration
Monitor
OSD
CephFS
Pools
Log
Replication
Create
Edit
Destroy
Name
Size/min
# of Placement Groups
Optimal # of PGs
Autoscale Mode
CRUSH Rule (ID)
Used (%)
device_health_metrics 3/2 1 1 on replicated_rule (0) 0 B (0.00%)
Group3-Pool 3/2 101 32 on replicated_rule (0) 0 B (0.00%)
0 B
Tasks
Cluster log
Start Time
End Time
Node
User name
Description
Status
May 21 13:29:57 Group3-No... root@pam Shell
May 21 13:31:52 May 21 13:32:00 Group3-No... root@pam Ceph Pool Group3-Pool - Create OK
May 21 13:31:19 May 21 13:31:26 Group3-No... root@pam Ceph OSD sda4 - Create OK
May 21 13:30:59 May 21 13:31:07 Group3-No... root@pam Ceph OSD sda4 - Create OK
May 21 13:30:39 May 21 13:30:46 Group3-No... root@pam Ceph OSD sda4 - Create OK
1:34 PM
5/21/2026

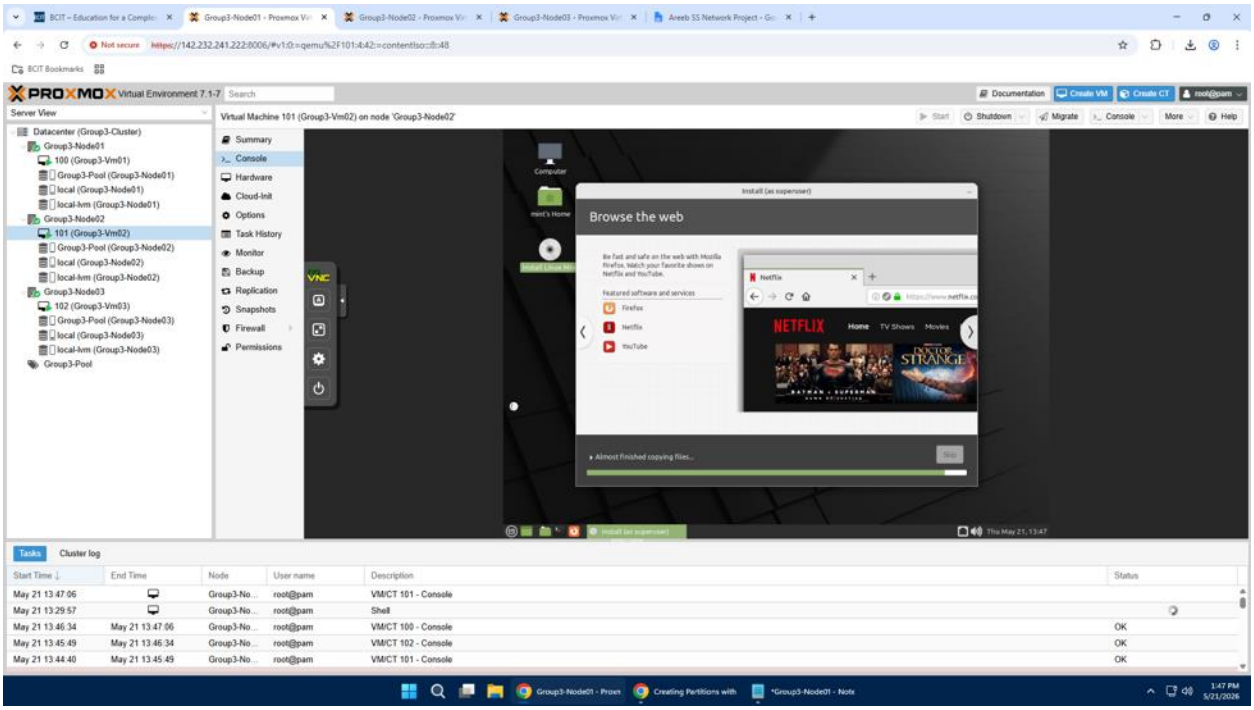

Group3-Node03 - Proxmox Virt
Not secure
https://142.232.241.220:8006/#v1:0:=node%2FGroup3-Node01:4:38:::5
PROXMOX Virtual Environment 7.1-7
Search
Documentation
Create VM
Create CT
root@pam
Server View
Node 'Group3-Node01'
Reboot
Shutdown
Shell
Bulk Actions
Help
Datacenter (Group3-Cluste
Group3-Node01
Group3-Node02
Group3-Node03
Group3-Pool
Search
Summary
Notes
Shell
System
Network
Certificates
DNS
Hosts
Time
Syslog
Updates
Repositories
Firewall
Disks
LVM
LVM-Thin
Directory
ZFS
Ceph
Health
Status
HEALTH_OK
Severity
Summary
No Warnings/Errors
Ceph Version:
16.2.15
Status
OSDs
In
Out
Up
3
0
Down
0
0
Total: 3
PGs
active+clean:
93
Services
Monitors
Managers
Meta Data Servers
Group3-Node01:
Group3-Node02:
Group3-Node03:
Performance
Reads:
0 B/s
Tasks
Cluster log
Start Time
End Time
Node
User name
Description
Status
May 21 13:29:57
Group3-No...
root@pam
Shell
May 21 13:31:52
May 21 13:32:00
Group3-No...
root@pam
Ceph Pool Group3-Pool - Create
OK
May 21 13:31:19
May 21 13:31:26
Group3-No...
root@pam
Ceph OSD sda4 - Create
OK
May 21 13:30:59
May 21 13:31:07
Group3-No...
root@pam
Ceph OSD sda4 - Create
OK
May 21 13:30:39
May 21 13:30:46
Group3-No...
root@pam
Ceph OSD sda4 - Create
OK
1:35 PM
5/21/2026

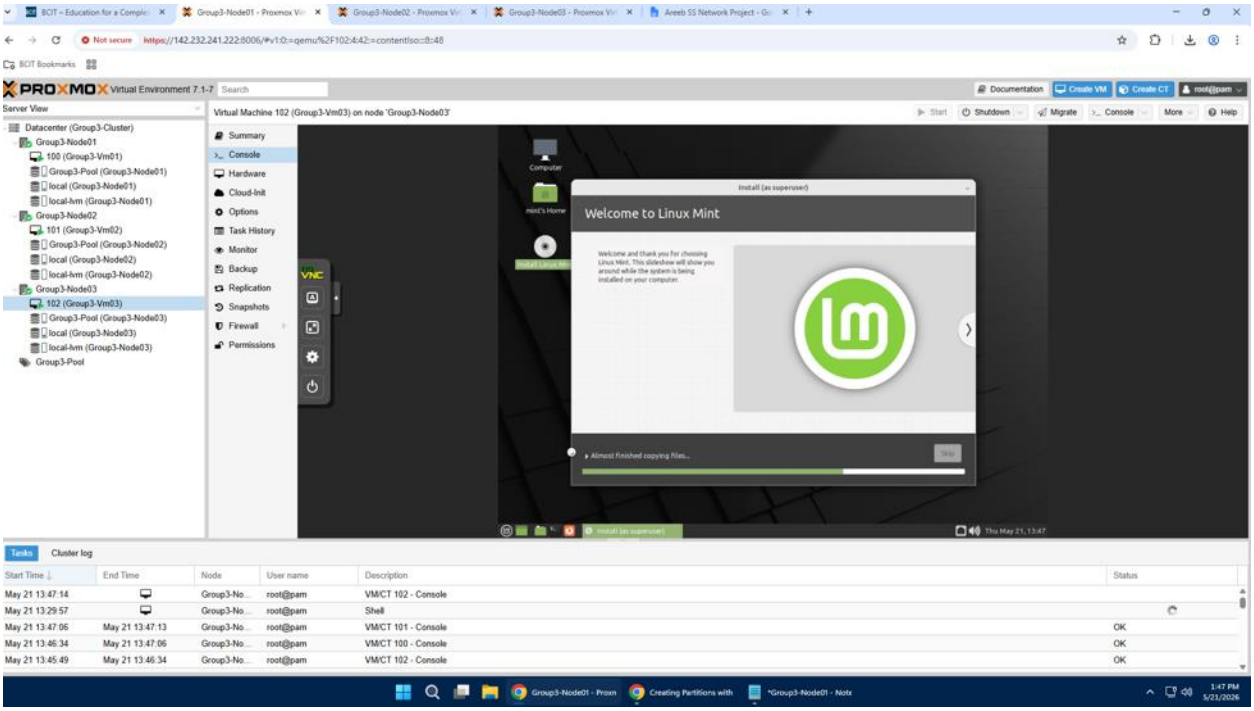

Group3-Node03 - Proxmox Virt
Not secure
https://142.232.241.220:8006/#v1:0:=node%2FGroup3-Node01:4:40:::5
PROXMOX Virtual Environment 7.1-7
Search
Documentation
Create VM
Create CT
root@pam
Server View
Node 'Group3-Node01'
Reboot
Shutdown
Shell
Bulk Actions
Help
Datacenter (Group3-Cluste
Group3-Node01
Group3-Node02
Group3-Node03
Group3-Pool
Time
Syslog
Updates
Repositories
Firewall
Disks
LVM
LVM-Thin
Directory
ZFS
Ceph
Configuration
Monitor
OSD
CephFS
Pools
Log
Replication
Task History
Subscription
Monitor
Start
Stop
Restart
Create
Destroy
Syslog
Name
Host
Status
Address
Version
Quorum
mon.Group3-Node01
Group3-Node01
running
142.232.241.222:6789/0
16.2.15
Yes
mon.Group3-Node02
Group3-Node02
running
142.232.241.221:6789/0
16.2.15
Yes
mon.Group3-Node03
Group3-Node03
running
142.232.241.220:6789/0
16.2.15
Yes
Manager
mgr.Group3-Node01
Group3-Node01
active
142.232.241.222
16.2.15
Tasks
Cluster log
Start Time
End Time
Node
User name
Description
Status
May 21 13:29:57
Group3-No...
root@pam
Shell
May 21 13:31:52
May 21 13:32:00
Group3-No...
root@pam
Ceph Pool Group3-Pool - Create
OK
May 21 13:31:19
May 21 13:31:26
Group3-No...
root@pam
Ceph OSD sda4 - Create
OK
May 21 13:30:59
May 21 13:31:07
Group3-No...
root@pam
Ceph OSD sda4 - Create
OK
May 21 13:30:39
May 21 13:30:46
Group3-No...
root@pam
Ceph OSD sda4 - Create
OK
1:36 PM

1.4 Installing Virtual Machines to each node

Each node has a virtual machine installed using Linux Mint OS with 16GB RAM and 500GB storage and with 8 CPU cores each.

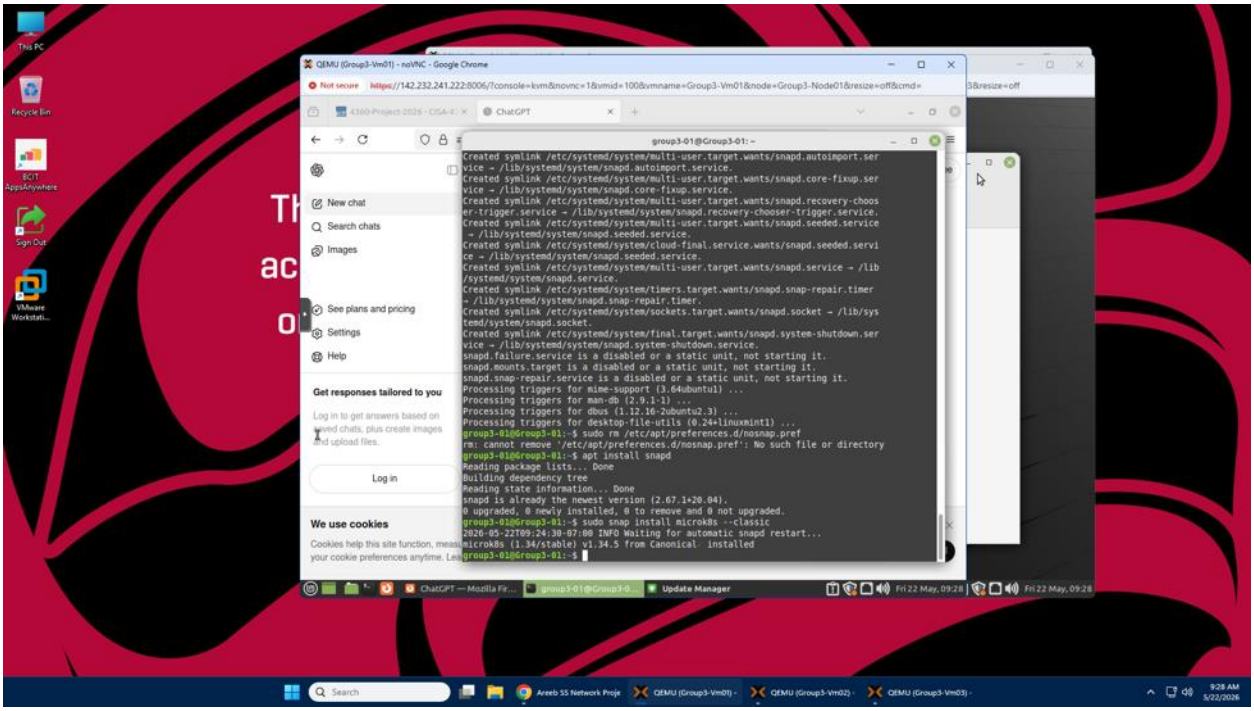


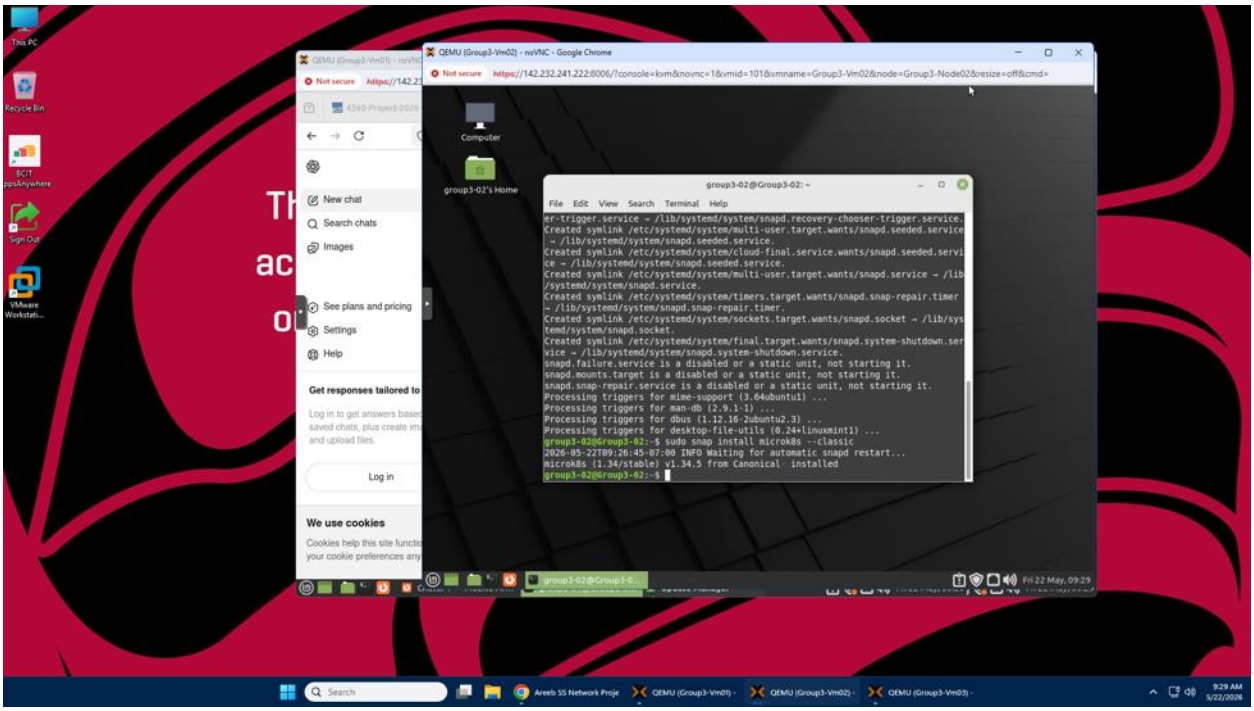


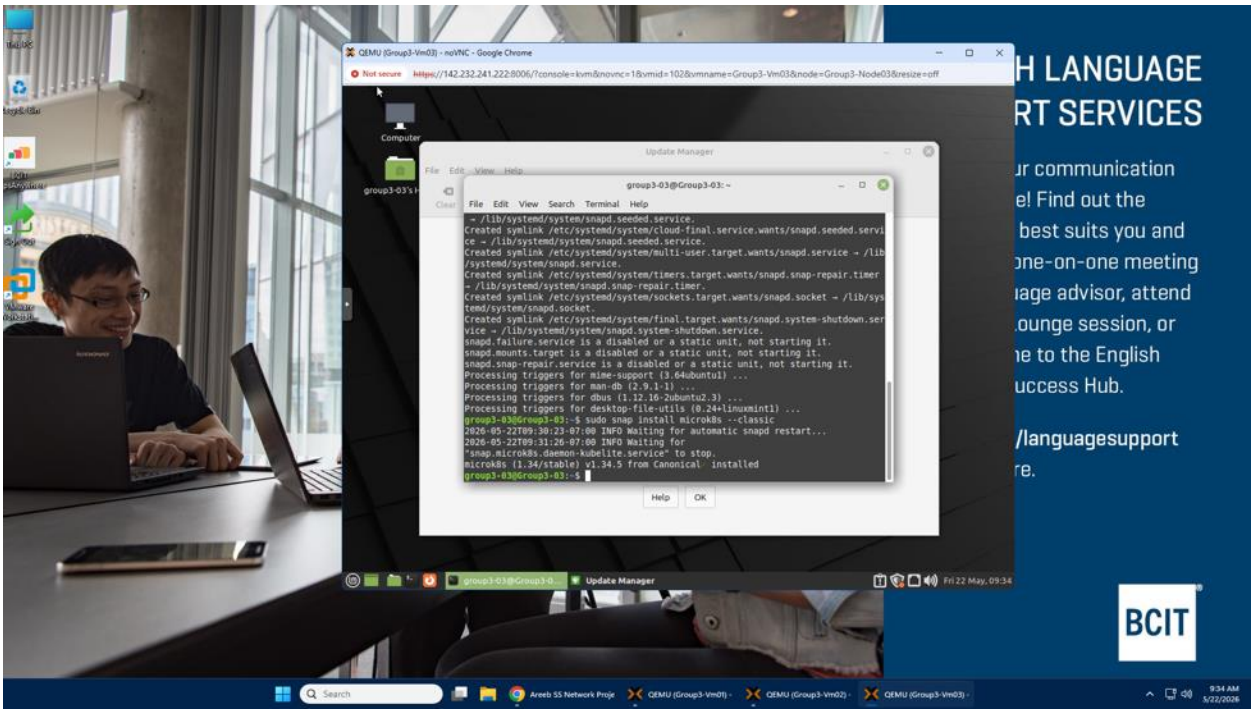

Virtual Environment 7.1-7
Server View
Datacenter (Group3-Cluster)
Group3-Node01
100 (Group3-Vm01)
Group3-Pool (Group3-Node01)
local (Group3-Node01)
local-lvm (Group3-Node01)
Group3-Node02
101 (Group3-Vm02)
Group3-Pool (Group3-Node02)
local (Group3-Node02)
local-lvm (Group3-Node02)
Group3-Node03
102 (Group3-Vm03)
Group3-Pool (Group3-Node03)
local (Group3-Node03)
local-lvm (Group3-Node03)
Group3-Pool
Virtual Machine 102 (Group3-Vm03) on node 'Group3-Node03'
Summary
Console
Hardware
Cloud-Init
Options
Task History
Monitor
Backup
Replication
Snapshots
Firewall
Permissions
Welcome to Linux Mint
Tasks
Cluster log
Start Time
End Time
Node
User name
Description
Status
May 21 13:47:14 Group3-No... root@pam VM/CT 102 - Console
May 21 13:29:57 Group3-No... root@pam Shell
May 21 13:47:06 May 21 13:47:13 Group3-No... root@pam VM/CT 101 - Console OK
May 21 13:46:34 May 21 13:47:06 Group3-No... root@pam VM/CT 100 - Console OK
May 21 13:45:49 May 21 13:46:34 Group3-No... root@pam VM/CT 102 - Console OK

Part 2: Kubernetes Installation

We installed MicroK8s for Kubernetes on each node as it is lightweight, simple and easy to setup.





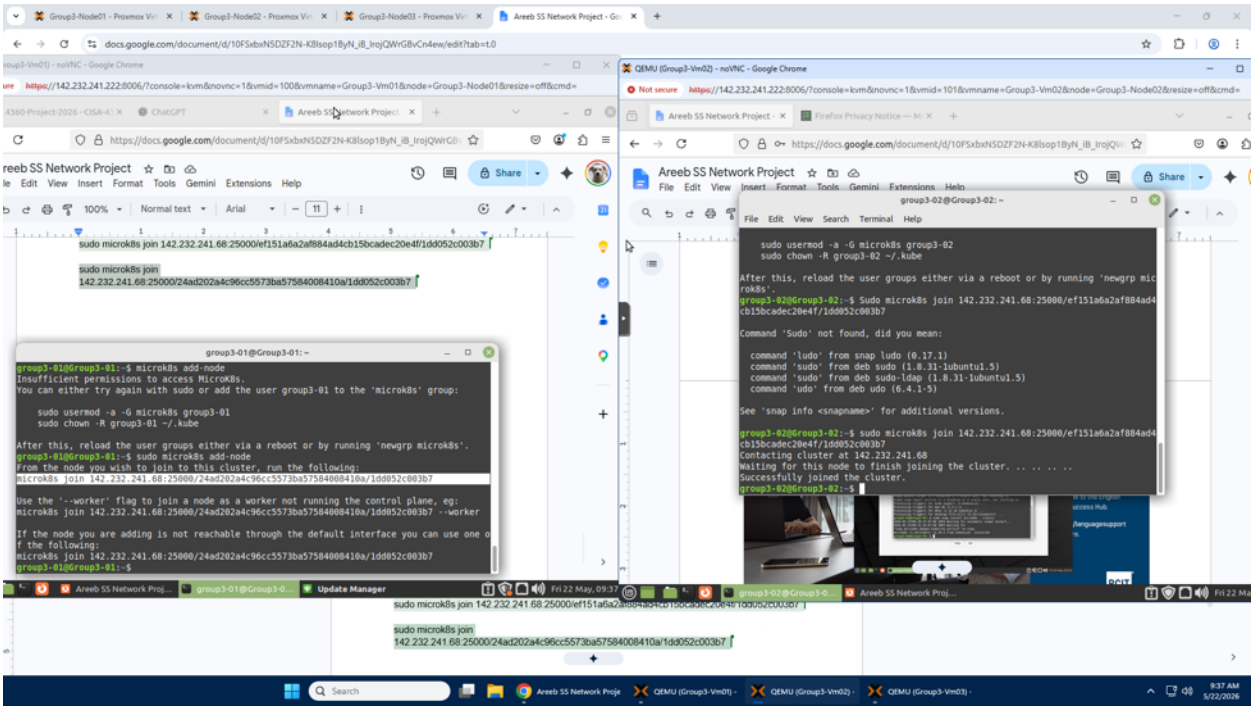


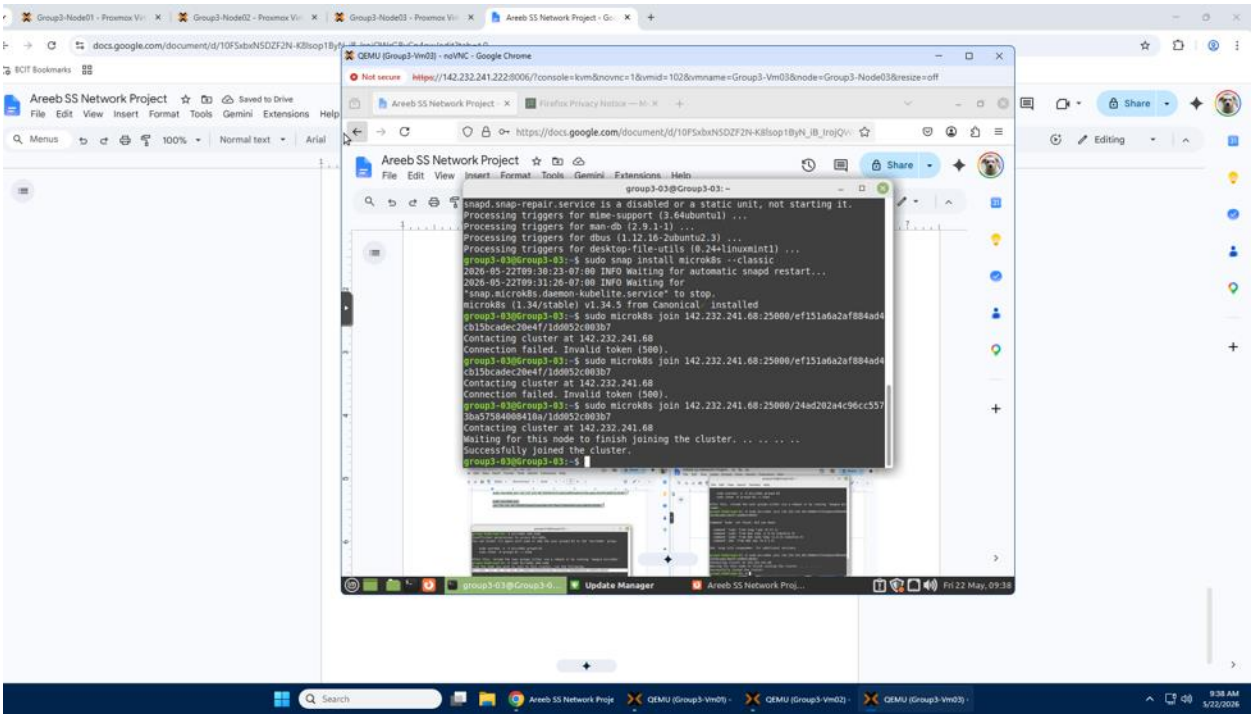
Commands used to install MicroK8s on all Virtual Machines

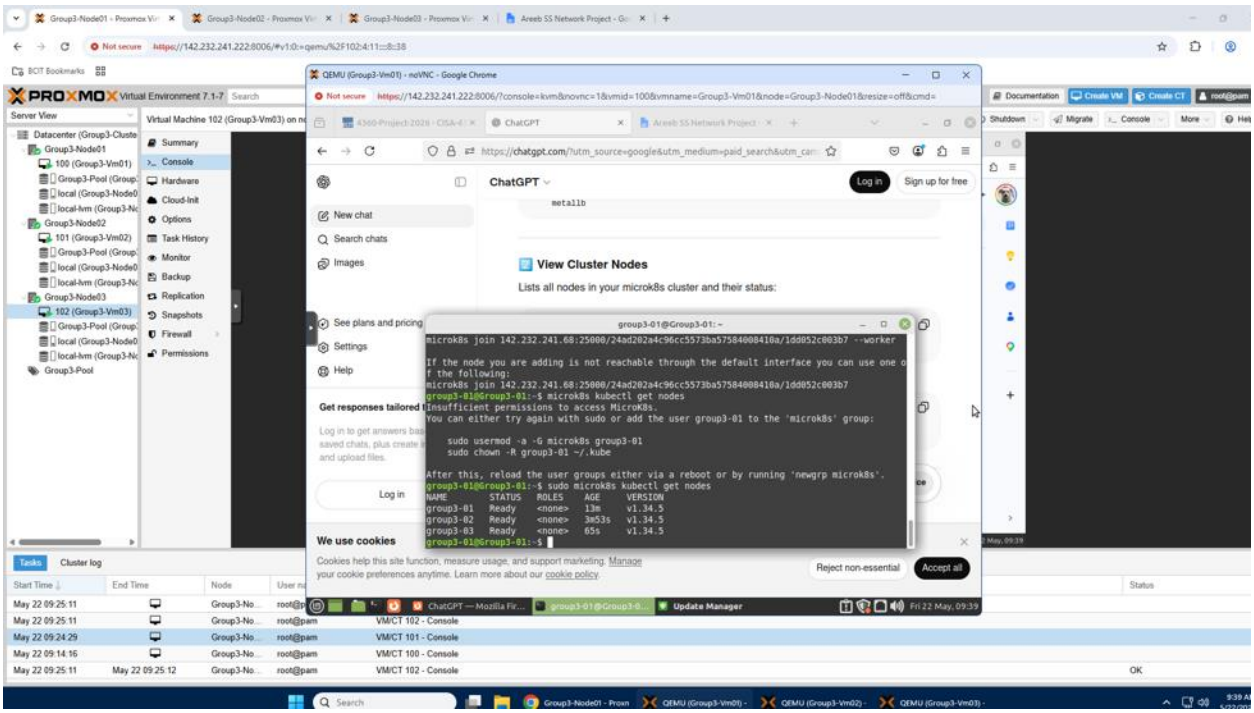
- **`sudo apt update` ->** to update the OS
- **`sudo apt upgrade` ->** to install the latest versions of already-installed packages
- **`sudo apt install snapd` ->** to install snapd
- **`sudo snap install microK8s –classic` ->** install microk8s using snap

2.1 Kubernetes Cluster

We create a Kubernetes cluster and configure the nodes to join the cluster for high availability.







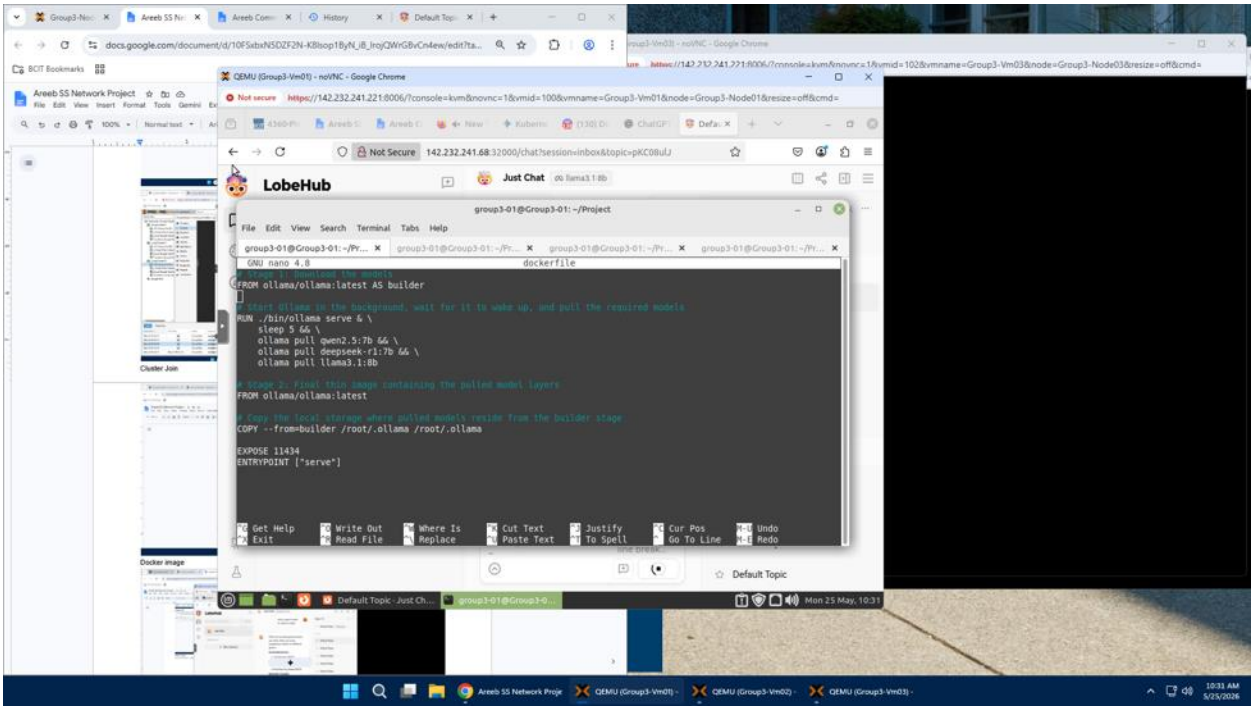
Commands used to create a cluster and join the cluster

- **`sudo microk8s add-node`** -> to create a cluster
- **`sudo microk8s join 142.232.241.68:2500...`** -> this the join command that needs to be copied and pasted to node 2 and node 3 to join the cluster
- **`sudo microk8s kubectl get nodes`** -> to view the status of each node

Part 3: AI Models Deployment, Lobechat and OpenClaw

We use Lobechat as the front-end interface and ollama, qwen, and deepseek as the local AI model backend. The AI models are deployed locally inside of our Kubernetes environment and AI responses are generated on our infrastructure instead of using cloud-hosted APIs

3.1 Creating a Docker File



Commands to create docker file and docker file configurations

- **`sudo snap install docker` ->** to install docker
- **`nano dockerfile` ->** command used to modify the docker file

Contents of the docker file

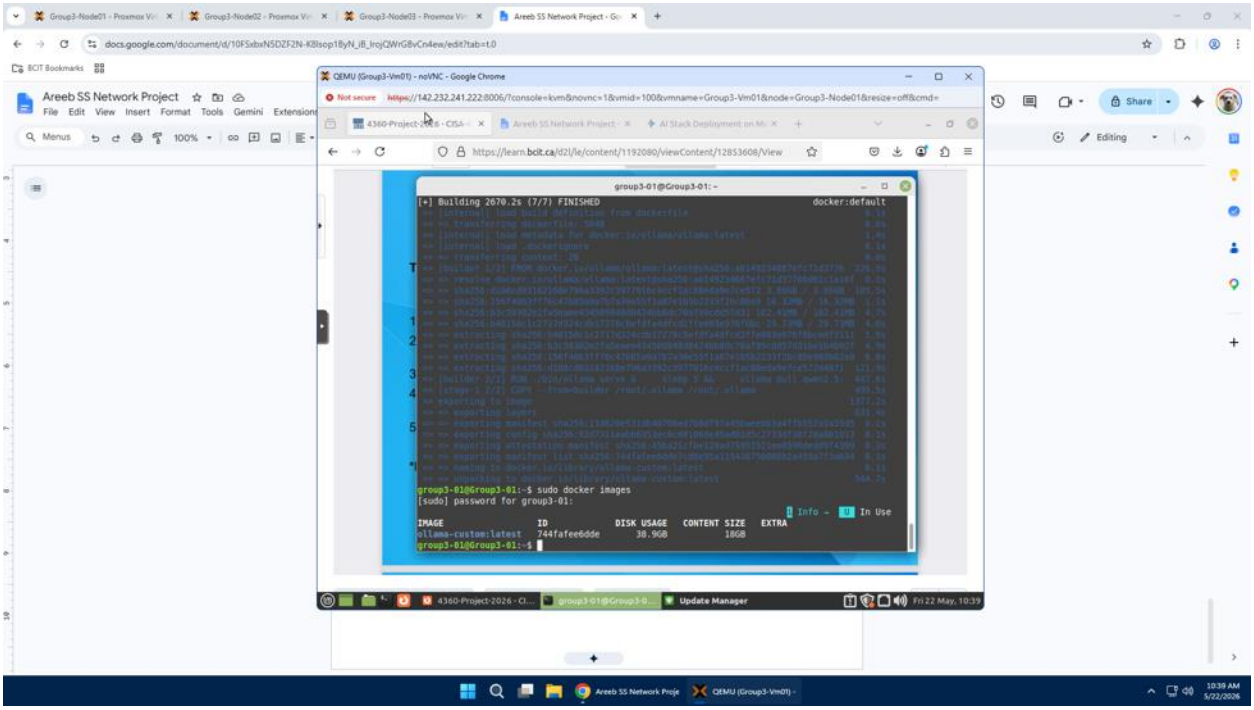
The docker configuration file that downloads the AI models and copies those files into the container image. And AI models will automatically start on port 11434.

Dockerfile

FROM ollama/ollama:latest AS builder
 	
RUN ./bin/ollama serve & \
  sleep 5 && \
  ollama pull qwen2.5:7b && \
  ollama pull deepseek-r1:7b && \
  ollama pull llama3.1:8b
 	
FROM ollama/ollama:latest
 	
COPY --from=builder /root/.ollama /root/.ollama

```
EXPOSE 11434
ENTRYPOINT ["serve"]
```

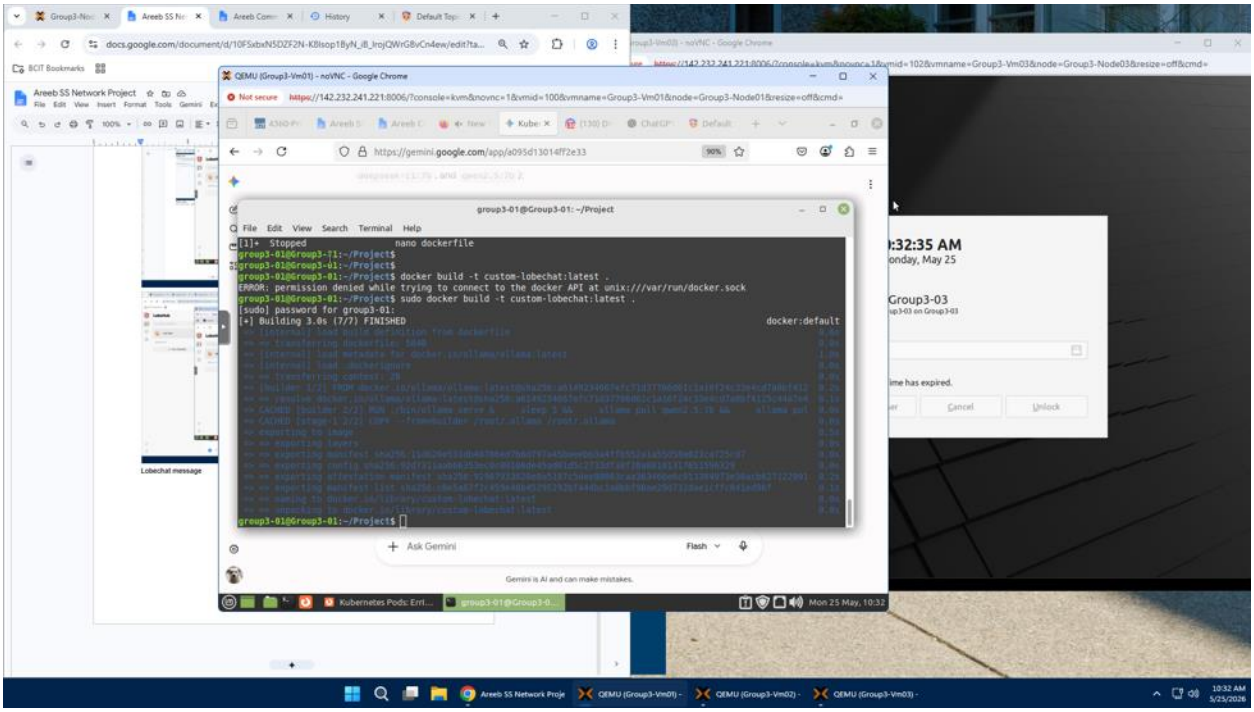
3.2 Docker Ollama Image Build



Commands used for the Ollama image deployment

- **`sudo docker build -t ollama-custom:latest .`** -> a command to build a docker image for ollama
- **`sudo docker save -o group3-ollama-image.tar ollama-image:latest`** -> a command to export docker image into a file
- **`sudo docker save ollama-custom:latest | gzip > ollama.tar.gz`** -> exporting the ollama-custom:latest and compressing it.
- **`sudo microk8s images import ollama.tar.gz`** -> imports docker images into MicroK8s internal container registry.
- **`sudo docker images`** -> to view the docker images

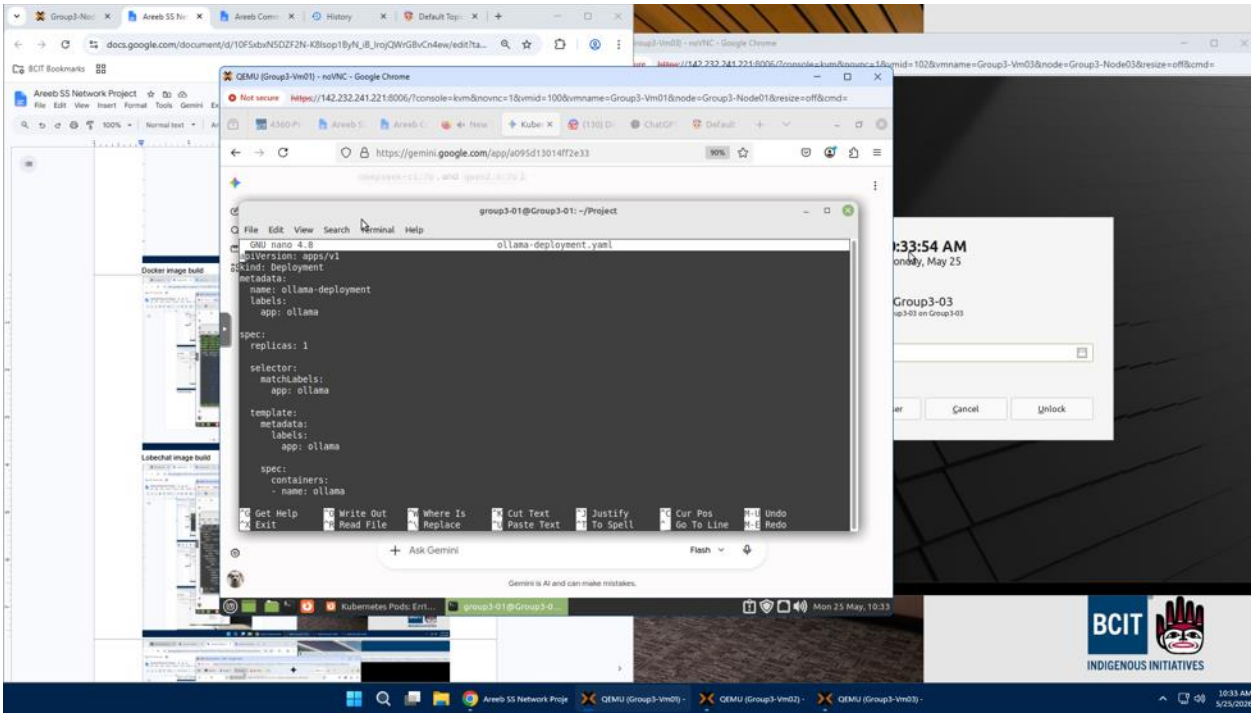
3.3 Lobechat/LobeHub Image build



Commands used for the Lobechat/LobeHub image deployment

- **`sudo docker build -t lobechat-custom:latest .`** -> a command to build a docker image for lobechat
- **`sudo docker save custom-lobechat:latest > custom-lobechat.tar ->`** exporting the custom-lobechat:latest and compressing it.
- **`sudo microk8s ctr images import custom-lobechat.tar ->`** imports the docker images directly into MicroK8s

3.4 Ollama Deployment in MicroK8s



Commands used for the deployment of Ollama in MicroK8s

- **`nano ollama-deployment.yaml`** -> to create and edit the ollama-deployment YAML file
- **`microk8s kubectl apply -f ollama-deployment.yaml`** -> to create and update resources using the yaml file inside Kubernetes (microK8s)
- **`microk8s kubectl get pods`** -> to display the active pods
- **`microk8s kubectl delete pod -l app= <app-name>` ->** to delete a pod

Contents of the ollama-deployment.yaml file

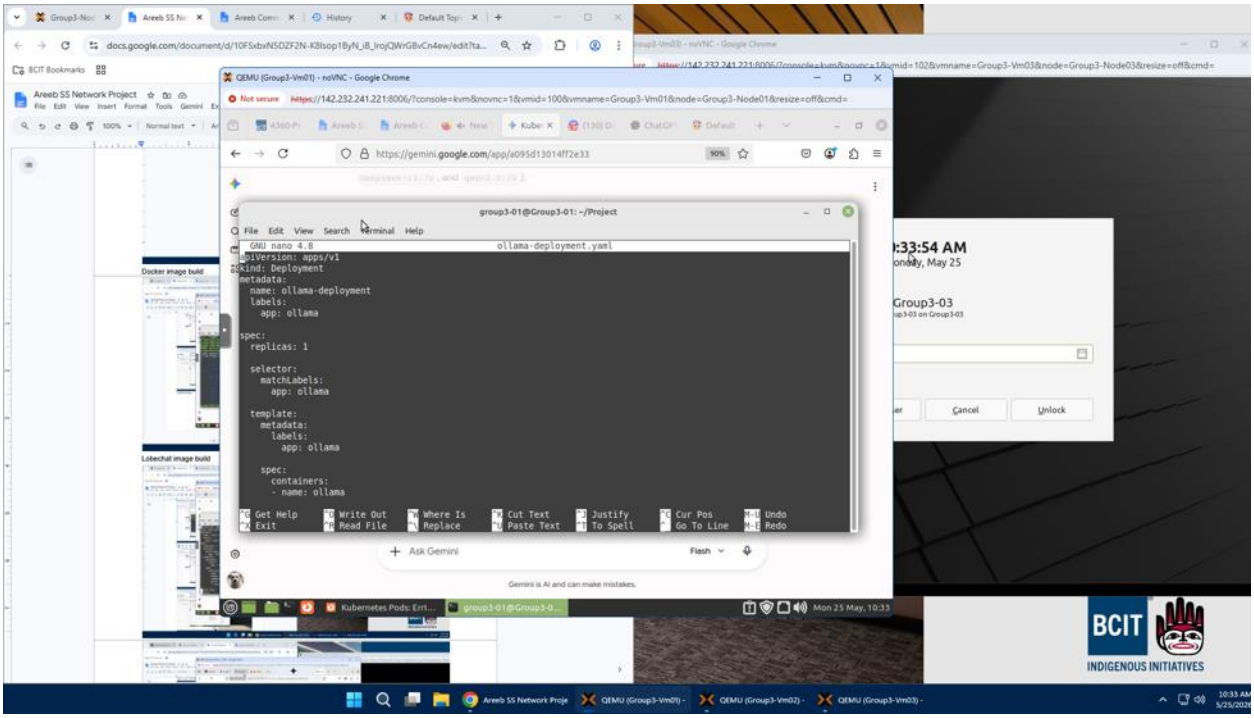
The YAML file that deploys ollama server to MicroK8s using the local image and runs it as a container and exposing it to port number 11434 that creates Kubernetes services where lobechat can access it within cluster.

ollama-deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: ollama-deployment
  labels:
  app: ollama
 	
spec:

```
  replicas: 1

  selector:
    matchLabels:
      app: ollama

  template:
    metadata:
      labels:
        app: ollama

    spec:
      containers:
      - name: ollama

        image: docker.io/library/ollama-custom:latest
        imagePullPolicy: Never

        command: ["/bin/sh", "-c"]
        args:
          - exec ollama serve
        env:

        - name: OLLAMA_ORIGINS
          value: "*"

        - name: OLLAMA_HOST
          value: "0.0.0.0:11434"

        ports:
        - containerPort: 11434

---
apiVersion: v1
kind: Service
metadata:
  name: ollama-service

spec:
  type: ClusterIP

  selector:
    app: ollama

  ports:
  - protocol: TCP
    port: 11434
    targetPort: 11434
```

3.5 LobeChat/LobeHub Deployment in MicroK8s



Commands used for the deployment of Ollama in MicroK8s

- **`nano lobechat-deployment.yaml`** -> to create and edit the lobechat-deployment YAML file
- **`microk8s kubectl apply -f lobechat-deployment.yaml`** -> to create and update resources using the yaml file inside Kubernetes (microK8s)
- **`microk8s kubectl get pods`** -> to display the active pods
- **`microk8s kubectl delete pod -l app= <app-name>` ->** to delete a pod

Contents of the lobechat-deployment.yaml file

The YAML file that deploys LobeChat as a frontend in Kubernetes and connects our local ollama service and exposing the chat app through nodeport so we can access it from the browser

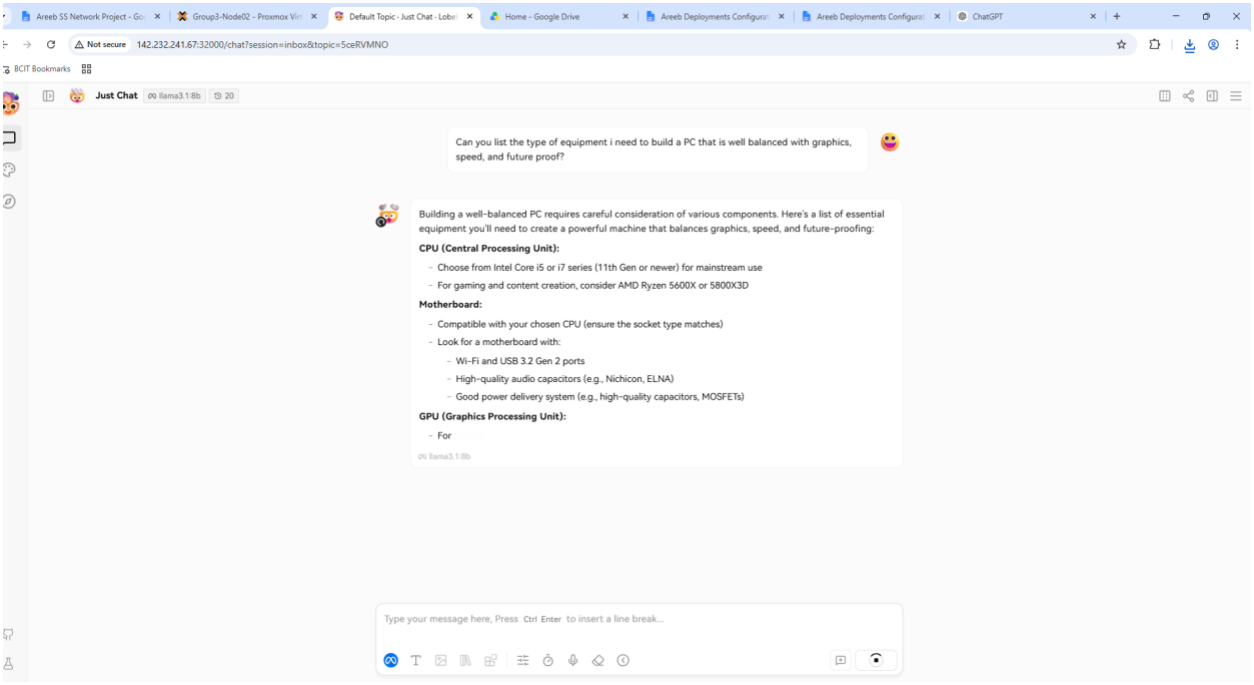
lobechat-deployment.yaml

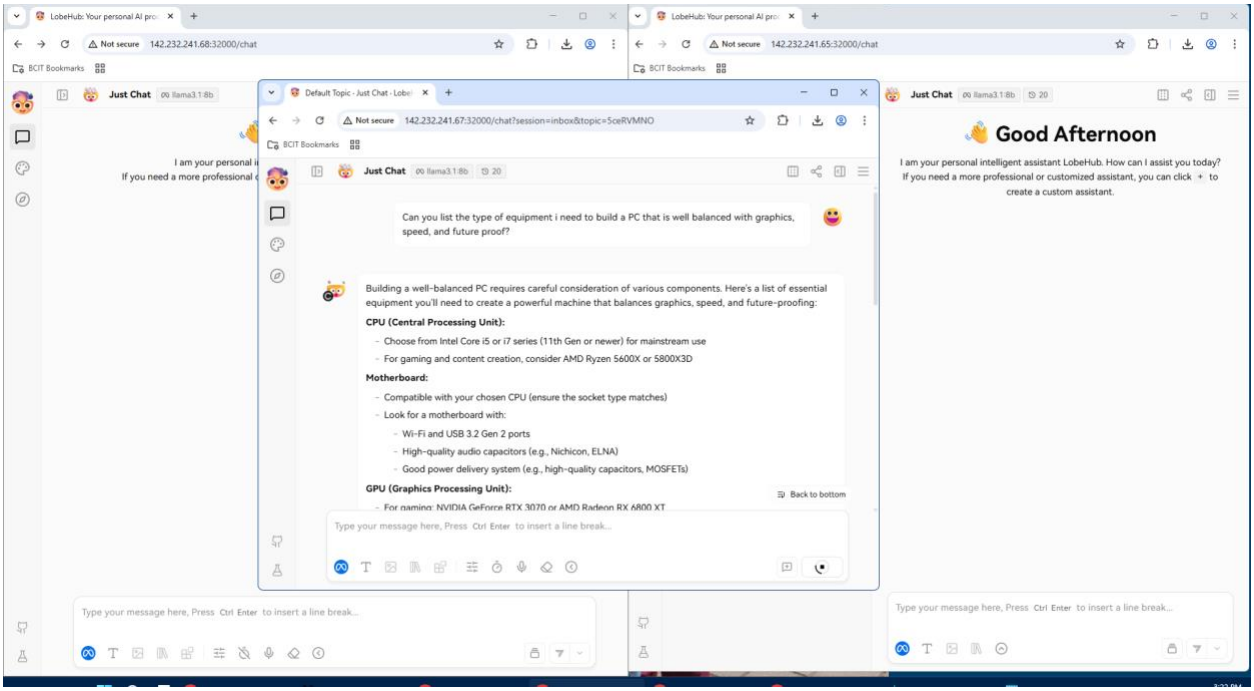
apiVersion: apps/v1
kind: Deployment
metadata:
  name: claw-lobechat-deployment
  labels:
  app: lobechat
spec:
  replicas: 1
  selector:

```
    matchLabels:
      app: lobechat
  template:
    metadata:
      labels:
        app: lobechat
    spec:
      containers:
      - name: lobe-chat
        image: docker.io/library/custom-lobechat:latest
        imagePullPolicy: Never
        ports:
        - containerPort: 3210
        env:
        # Routes backend requests safely via core Kubernetes DNS proxying
        - name: OLLAMA_PROXY_URL
          value: "http://ollama-service:11434"
        - name: OLLAMA_PROXY
          value: "1"
        - name: NEXT_PUBLIC_OLLAMA_PROXY_URL
          value: "http://ollama-service:11434"
        # Maps names to tags natively so LobeChat won't drop the ':8b' or ':7b' suffixes
        - name: CUSTOM_MODELS
          value: '-all,+llama3.1=llama3.1:8b,+deepseek-r1=deepseek-r1:7b,+qwen2.5=qwen2.5:7b'
        - name: DISABLE_IMAGE_OPTIMIZATION
          value: "1"
        - name: OPENCLAW_PROXY_URL
          value: "http://openclaw-service.default.svc.cluster.local:18789"
        - name: OPENCLAW_API_KEY
          value: ollama-local
---
apiVersion: v1
kind: Service
metadata:
  name: lobechat-nodeport
spec:
  type: NodePort
  selector:
    app: lobechat
  ports:
  - port: 3210
    targetPort: 3210
```

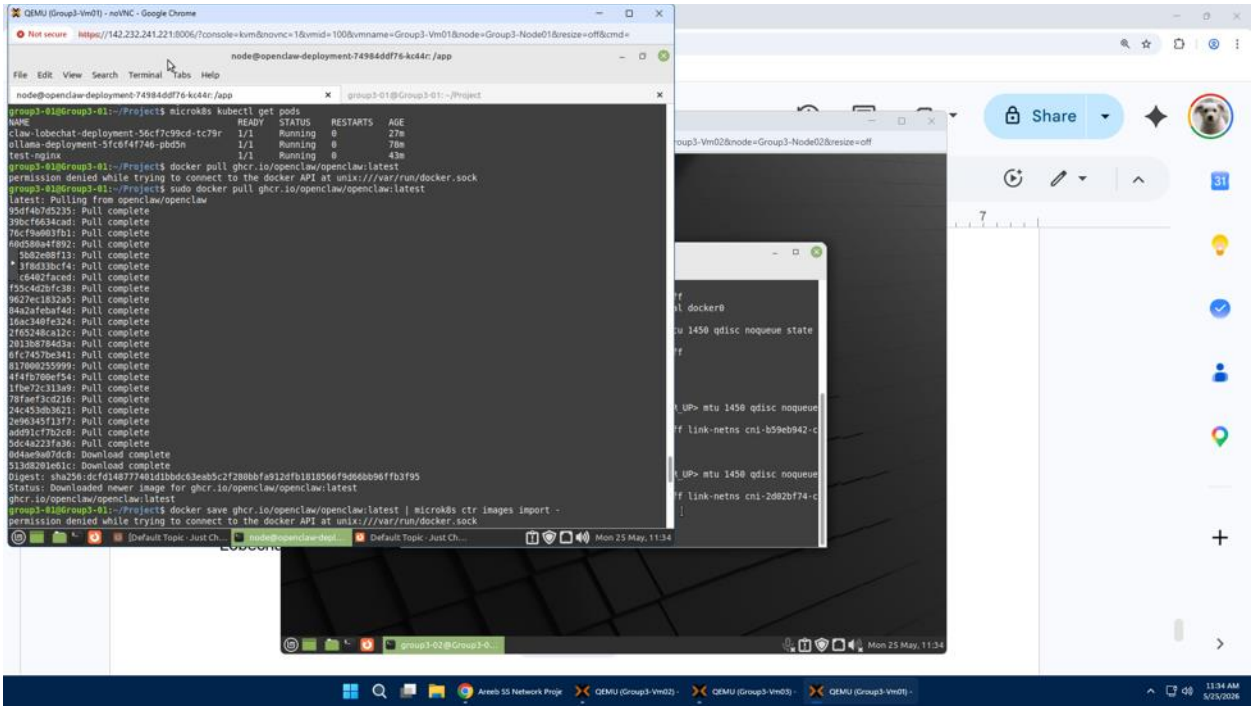
3.6 Lobechat GUI

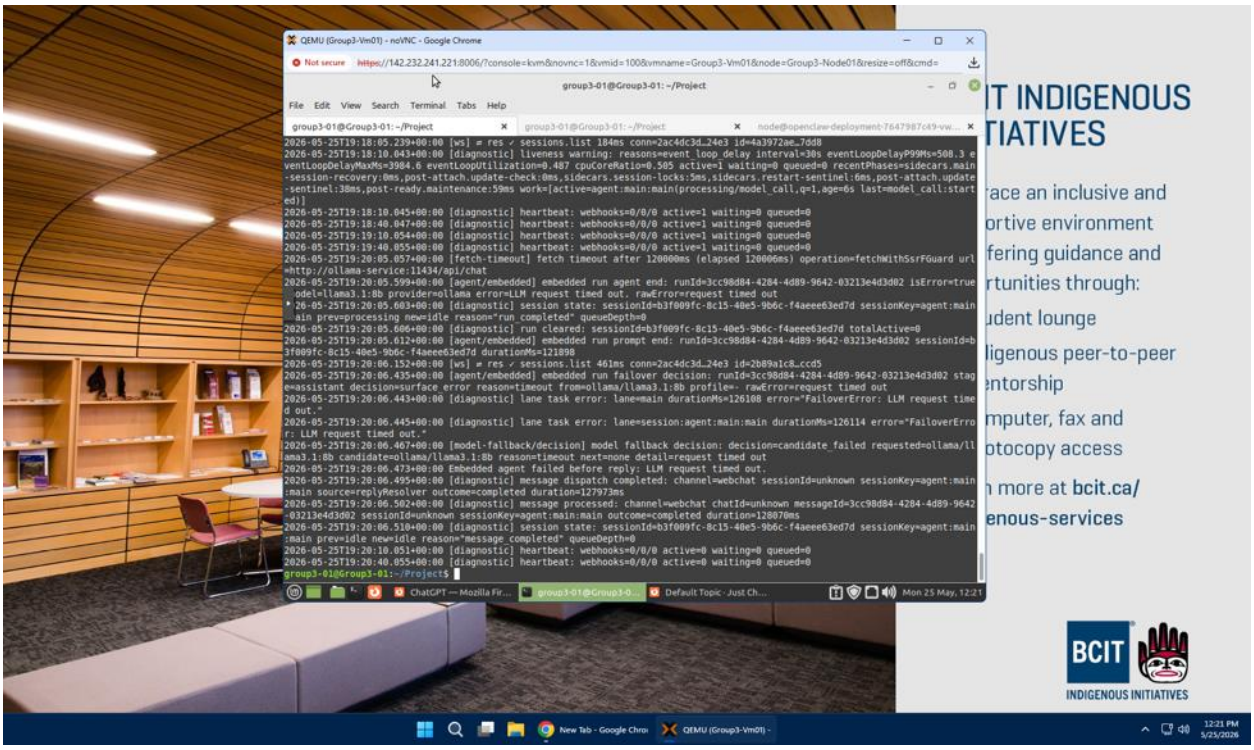
Lobechat/lobehub is accessible via web browser using VM's IP address and exposed to port number 32000. And able to answer/respond to our queries. For high availability, the LobeChat/LobeHub is accessible using the other cluster Node's IP addresses to ensure the continued high availability even if the one node becomes unavailable.

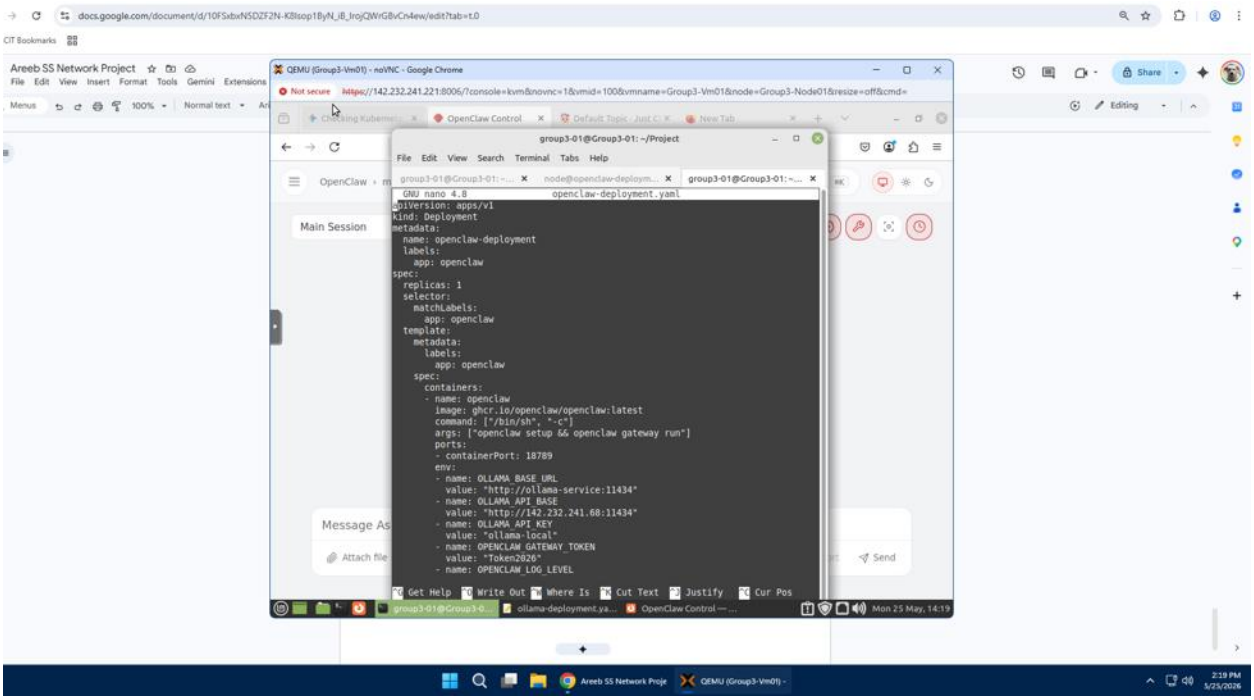




3.7 OpenClaw deployment in MicroK8s







Commands used for the deployment of OpenClaw in MicroK8s

- **`nano openclaw-deployment.yaml`** -> to create and edit the openclaw-deployment YAML file
- **`microk8s kubectl apply -f openclaw-deployment.yaml`** -> to create and update resources using the yaml file inside Kubernetes (microK8s)
- **`microk8s kubectl port-forward pods/openclaw-deployment-7647987c49-jntxs 18789:18789 ->`** to expose the openclaw using the port 18789
- **`microk8s kubectl exec –it openclaw-deployment-7647987c49-jntxs -- /bin/bash ->`** to enter to OpenClaw terminal.

Contents of the openclaw-deployment.yaml file

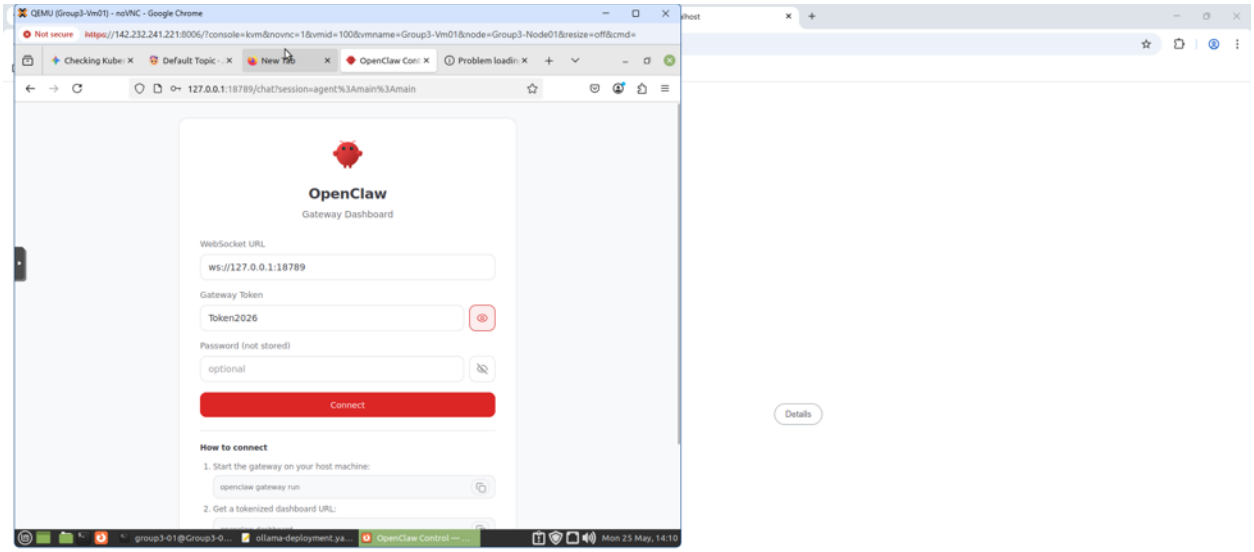
The YAML file that deploys OpenClaw service in Kubernetes using microK8s. That runs in one container and connects to ollama as AI backend and it starts the gateway in port 18789.

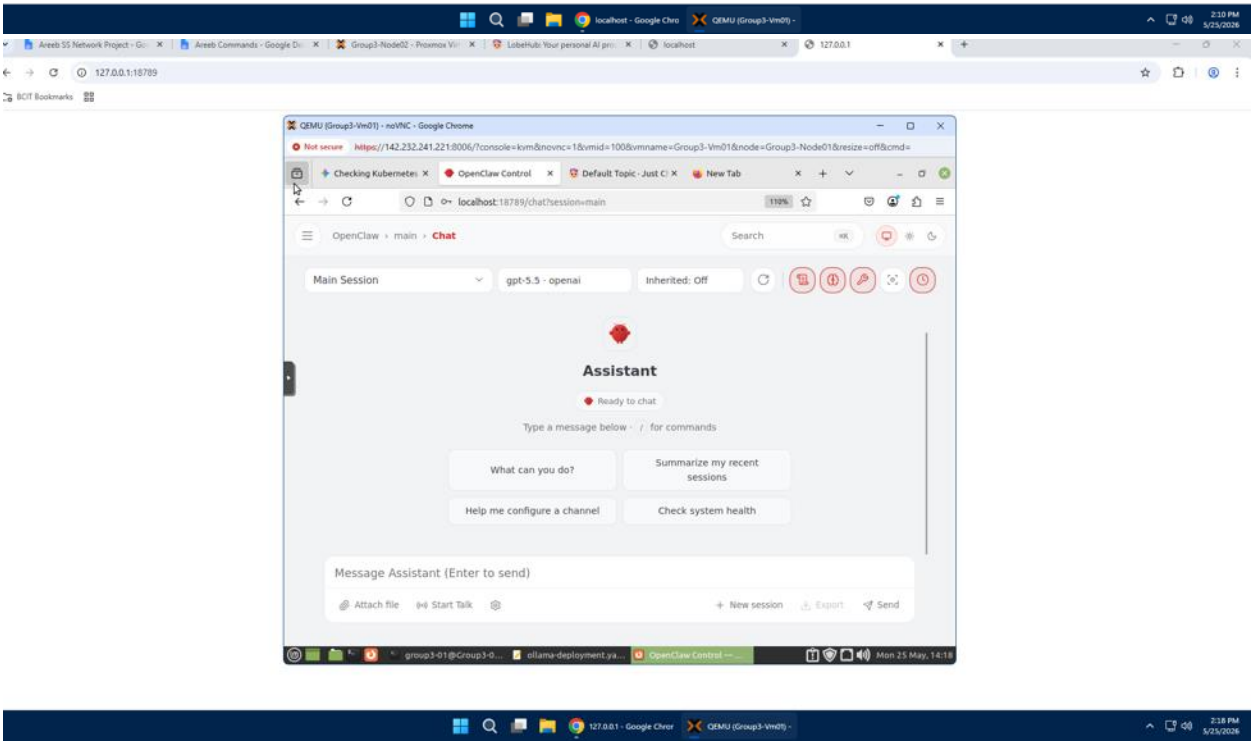
openclaw-deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: openclaw-deployment
  labels:
  app: openclaw
spec:
  replicas: 1
  selector:
  matchLabels:

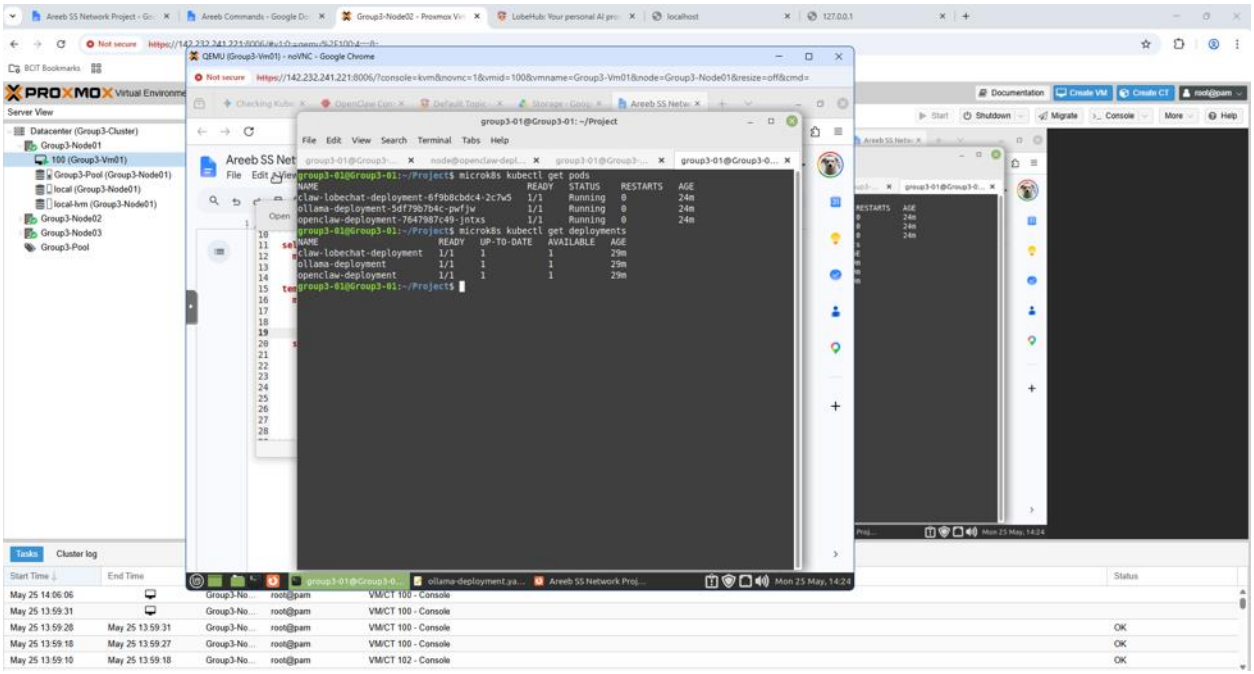
```
      app: openclaw
  template:
    metadata:
      labels:
        app: openclaw
    spec:
      containers:
      - name: openclaw
        image: ghcr.io/openclaw/openclaw:latest
        command: ["/bin/sh", "-c"]
        args: ["openclaw setup && openclaw gateway run"]
        ports:
        - containerPort: 18789
        env:
        - name: OLLAMA_BASE_URL
          value: "http://ollama-service:11434"
        - name: OLLAMA_API_BASE
          value: "http://142.232.241.68:11434"
        - name: OLLAMA_API_KEY
          value: "ollama-local"
        - name: OPENCLAW_GATEWAY_TOKEN
          value: "Token2026"
---
apiVersion: v1
kind: Service
metadata:
  name: openclaw-service
spec:
  selector:
    app: openclaw
  ports:
  - protocol: TCP
    port: 18789
    targetPort: 18789
```

3.8 OpenClaw GUI





3.9 All deployments and Pods



Commands used view deployments and pods

- **`microk8s kubectl get pods`** -> to display to running pods
- **`microk8s kubectl get deployments`** -> to display the active deployments

Other commands used

- **`systemctl status microk8s` ->** to check the status of microk8s
- **`microk8s stop` ->** to stop microk8s
- **`microk8s start` ->** to start microk8s
- **`ls -l` ->** to list all the files and folders in long listing format
- **`curl –I <address>:<port>` -** to check if the URL is accessible
- **`ifconfig`** -> to check the ip addresses
- **`history`** -> to display the used commands

Pros and Cons of the Infrastructure

The pros are the high availability even if the server failed. It is fully open-source and free, and using local AI gives us better privacy. And scalability is easier using Kubernetes; it has automated deployments using Docker and Kubernetes files. This project is very educational and great as real-world experience

The cons are the hardware resources limitations since we are doing this using classroom hardware/machines where no GPUs installed on it, the AI responses are much slower because AI models use a lot of storage and system resources and we only rely on CPU and RAM for performance. The Ceph and Kubernetes are difficult to configure and troubleshoot.

Could AI truly do your whole project and where do humans may be needed?

No, but AI helped us in generating code, creating Docker files and Kubernetes files. It helps us understand the flow of this project. It also helps us in troubleshooting since deploying applications in docker and Kubernetes is complicated.

Human interaction is still needed in setting up the hardware, networking, and manually installing all the required software for this project. Humans still need to fix and troubleshoot infrastructure problems. Human interaction is still needed for testing and validating the system like confirming the cluster stability and confirming that the failover functionality is working properly and ensuring that the networking is configured properly and all the nodes can communicate. Humans are still responsible for decision making, not AI.

This project is great for a real-world learning experience, and it makes us realize how powerful and helpful AI is, that accelerates the progress of our project especially in giving us the steps and what applications/software we need, generating codes and troubleshooting. But AI is not a replacement for system administrators or network administrators. AI only helps us to improve productivity but for the successful implementation of this project it still needs human supervision and technical knowledge. System/network administrators are still necessary to build, manage, troubleshoot, and maintain the infrastructure.