

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS,
ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES



COMUNICACIONES AVANZADAS

Octavo 'A'

CONSULTA

Tema:

KUBERNETS

Integrantes:

Yugcha Edison

Fecha de Envío:

Miercoles 25 de Octubre 2023

Fecha de Entrega:

Jueves 26 de Octubre de 2023

Docente: Ing. Mg. Santiago Manzano

Septiembre 2023 - Febrero 2024

AMBATO-ECUADOR

2023

Índice

1. TEMA	2
2. OBJETIVOS	2
2.1. Objetivo General	2
2.2. Objetivos Específicos	2
3. Resumen	2
4. FUNDAMENTACIÓN TEÓRICA	2
4.1. Alma Linux	2
4.2. Kubernetes	2
4.3. Clústeres de Kubernetes	3
4.4. Nodo Kubernetes	4
4.5. INSTALACION DE KUBERNET	4
4.6. Desarrollo	4
5. CONCLUSIONES	13
6. BIBLIOGRAFÍA	14
7. ANEXOS	14

Índice de figuras

1. Kubernetes	3
2. Cluster Kubernetes	3
3. Nodos Kubernetes	4
4. Actualizamos el repositorio	4
5. repositorio EPEL	4
6. Instalamos snapd	5
7. compatibilidad de complementos	5
8. Exportar las instantáneas Path	5
9. Iniciamos y habilitamos el servicio	6
10. modo permisivo	6
11. Instalamos Mikro8s	7
12. permisos	7
13. Ingresamos los comandos	7
14. Nodos Disponibles	8
15. versión de kubectl	8
16. configuraciones debidas para las sentencias	8
17. Instalamos kubectl en Rocky Linux 9/AlmaLinux 9	9
18. Damos los permisos	9
19. Lo movemos a donde lo necesitamos	9
20. configuración necesaria	9
21. nodos disponibles	9
22. otra maquina virtual	9
23. nodo MASTER	10
24. direccion ip del nodo esclavo	10
25. comando que utilizaran los nodos para unirse al nodo master	11
26. digitamos el siguiente código	11
27. puertos que vamos a necesitar	12
28. Añadimos la dirección	12
29. comando generado	13
30. comando generado	13
31. ANEXO 1	14
32. ANEXO 2	15
33. ANEXO 3	16

1. TEMA

Utilizar Kubernetes en Alma Linux

2. OBJETIVOS

2.1. Objetivo General

- Aplicar los conceptos básicos de Kubernetes para la automatización y administración de nodos

2.2. Objetivos Específicos

- Comprender la arquitectura de Kubernetes y sus componentes básicos.
- Realizar la instalacion de Kubernetes en Alma Linux .
- Para el desarrollo de la practica realizar la creacion de dos nodos el primero como nodo maestro y el segundo como nodo esclavo .

3. Resumen

Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios. Kubernetes facilita la automatización y la configuración declarativa. Tiene un ecosistema grande y en rápido crecimiento. El soporte, las herramientas y los servicios para Kubernetes están ampliamente disponibles.

Google liberó el proyecto Kubernetes en el año 2014. Kubernetes se basa en la experiencia de Google corriendo aplicaciones en producción a gran escala por década y media, junto a las mejores ideas y prácticas de la comunidad.

Kubernetes ofrece un entorno de administración centrado en contenedores. Kubernetes orquesta la infraestructura de cómputo, redes y almacenamiento para que las cargas de trabajo de los usuarios no tengan que hacerlo. Esto ofrece la simplicidad de las Plataformas como Servicio (PaaS) con la flexibilidad de la Infraestructura como Servicio (IaaS) y permite la portabilidad entre proveedores de infraestructura.

4. FUNDAMENTACIÓN TEÓRICA

4.1. Alma Linux

AlmaLinux OS es un sistema operativo Linux de código abierto, dirigido por la comunidad que llena el vacío dejado por la interrupción de la versión estable de CentOS Linux. AlmaLinux OS es una distribución Linux Enterprise, compatible con RHEL®, y guiado y construido por la comunidad.

Como sistema operativo independiente y completamente gratuito, el sistema operativo AlmaLinux disfruta de 1 millón de dólares en patrocinio anual de CloudLinux Inc. y el apoyo de más de otros 25 patrocinadores. Los esfuerzos de desarrollo en curso están gobernados por los miembros de la comunidad.

La Fundación AlmaLinux OS es una organización sin ánimo de lucro creada conforme a la legislación 501(c)(6) para beneficio de la comunidad de AlmaLinux OS.

4.2. Kubernetes

Kubernetes, también conocido como "k8s." "kube", es una plataforma de orquestación de contenedores para planificar y automatizar la implementación, gestión y escalamiento de aplicaciones en múltiples contenedores. Kubernetes fue desarrollado por primera vez por ingenieros en Google antes de pasarse a código abierto en 2014. Es un descendiente de "Borg", una plataforma de orquestación de contenedores utilizada internamente en Google. Kubernetes proviene de la palabra griega para timonel o piloto, de ahí el timón en el logotipo de Kubernetes (enlace externo a [ibm.com](https://kubernetes.io)).



kubernetes

Figura 1: Kubernetes

4.3. Clústeres de Kubernetes

Puede agrupar conjuntos de hosts que ejecuten contenedores de Linux® en clústeres, y Kubernetes lo ayudará a gestionarlos con facilidad y eficacia.

Los clústeres de Kubernetes pueden contener hosts en las instalaciones y en las nubes públicas, privadas o híbridas. Por eso, es la plataforma ideal para alojar las aplicaciones desarrolladas directamente en la nube que deben ajustarse rápidamente, como la transmisión inmediata de datos a través de Apache Kafka.

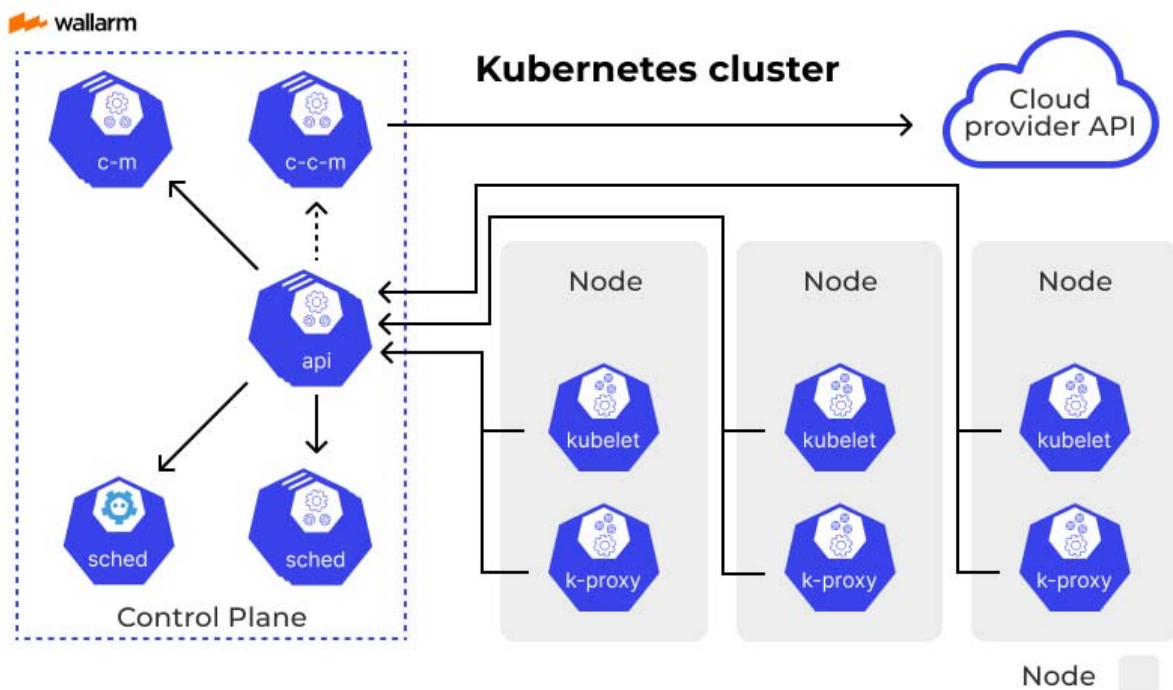


Figura 2: Cluster Kubernet

4.4. Nodo Kubernetes

Un nodo es una máquina de trabajo en Kubernetes, previamente conocida como minion. Un nodo puede ser una máquina virtual o física, dependiendo del tipo de clúster. Cada nodo está gestionado por el componente máster y contiene los servicios necesarios para ejecutar pods. Los servicios en un nodo incluyen el container runtime, kubelet y el kube-proxy. Accede a la sección The Kubernetes Node en el documento de diseño de arquitectura para más detalle.

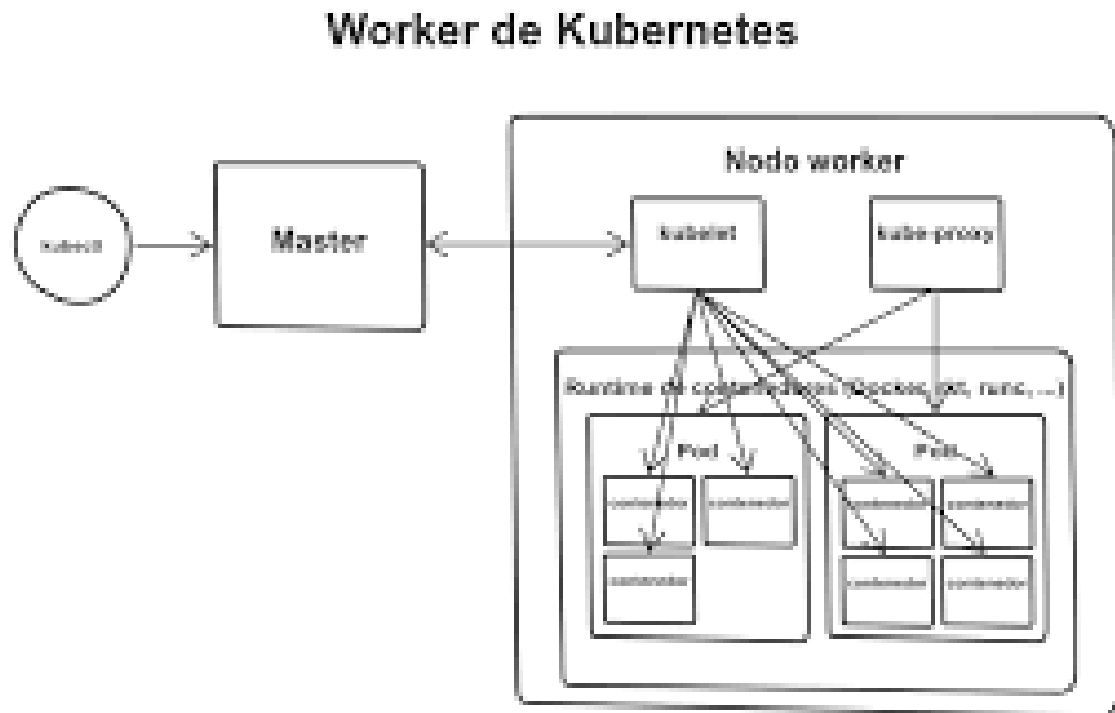


Figura 3: Nodos Kubernetes

4.5. INSTALACION DE KUBERNET

4.6. Desarrollo

1. Actualizamos el repositorio

```
root@localhost ~]# sudo dnf update
```

Figura 4: Actualizamos el repositorio

2. Instalamos el repositorio EPEL

```
root@localhost ~]# sudo dnf install epel-release
Last metadata expiration check: 1:59:17 ago on Tue Oct 24 14:02:19 2023.
Package epel-release-9-7.el9.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
root@localhost ~]#
```

Figura 5: repositorio EPEL

3. Instalamos snapd

```
root@localhost:~  
Installing:  
snapd                x86_64      2.58.3-1.el9      epel      15 M  
Installing dependencies:  
bash-completion      noarch      1:2.11-4.el9      baseos    291 k  
libpkgconf            x86_64      1.7.3-10.el9      baseos    35 k  
pkgconf              x86_64      1.7.3-10.el9      baseos    40 k  
pkgconf-m4           noarch      1.7.3-10.el9      baseos    14 k  
pkgconf-pkg-config    x86_64      1.7.3-10.el9      baseos    9.9 k  
snap-confine          x86_64      2.58.3-1.el9      epel      2.5 M  
snapd-selinux         noarch      2.58.3-1.el9      epel      192 k  
  
Transaction Summary  
=====  
Install 8 Packages  
  
Total download size: 18 M  
Installed size: 60 M  
Is this ok [y/N]: y  
Downloading Packages:  
1/8): libpkgconf-1.7.3-10.el9.x86_64.rpm      106 kB/s | 35 kB      00:00  
2/8): pkgconf-m4-1.7.3-10.el9.noarch.rpm      82 kB/s | 14 kB      00:00
```

Figura 6: Instalamos snapd

4. Enlace simbólico para la compatibilidad de complementos

```
root@localhost ~]# sudo ln -s /var/lib/snapd/snap /snap  
root@localhost ~]#
```

Figura 7: compatibilidad de complementos

5. Exportar las instantáneas Path

```
[root@localhost ~]# echo 'export PATH=$PATH:/var/lib/snapd/snap/bin' | sudo tee  
-a /etc/profile.d/snap.sh source /etc  
tee: /etc: Is a directory  
'export PATH=/root/.local/bin:/root/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin  
:/usr/bin:/var/lib/snapd/snap/bin'  
[root@localhost ~]#
```

Figura 8: Exportar las instantáneas Path

6. Iniciamos y habilitamos el servicio

```

root@localhost ~]# sudo systemctl enable --now snapd.socket
Created symlink /etc/systemd/system/sockets.target.wants/snapd.socket → /usr/lib/systemd/system/snapd.socket.
root@localhost ~]# systemctl status snapd.socket

● snapd.socket - Socket activation for snappy daemon
   Loaded: loaded (/usr/lib/systemd/system/snapd.socket; enabled; preset: disabled)
   Active: active (listening) since Tue 2023-10-24 16:09:06 -05; 24s ago
     Until: Tue 2023-10-24 16:09:06 -05; 24s ago
  Triggers: ● snapd.service
    Listen: /run/snapd.socket (Stream)
           /run/snapd-snap.socket (Stream)
     Tasks: 0 (limit: 22937)
    Memory: 0B
       CPU: 1ms
    CGroup: /system.slice/snapd.socket

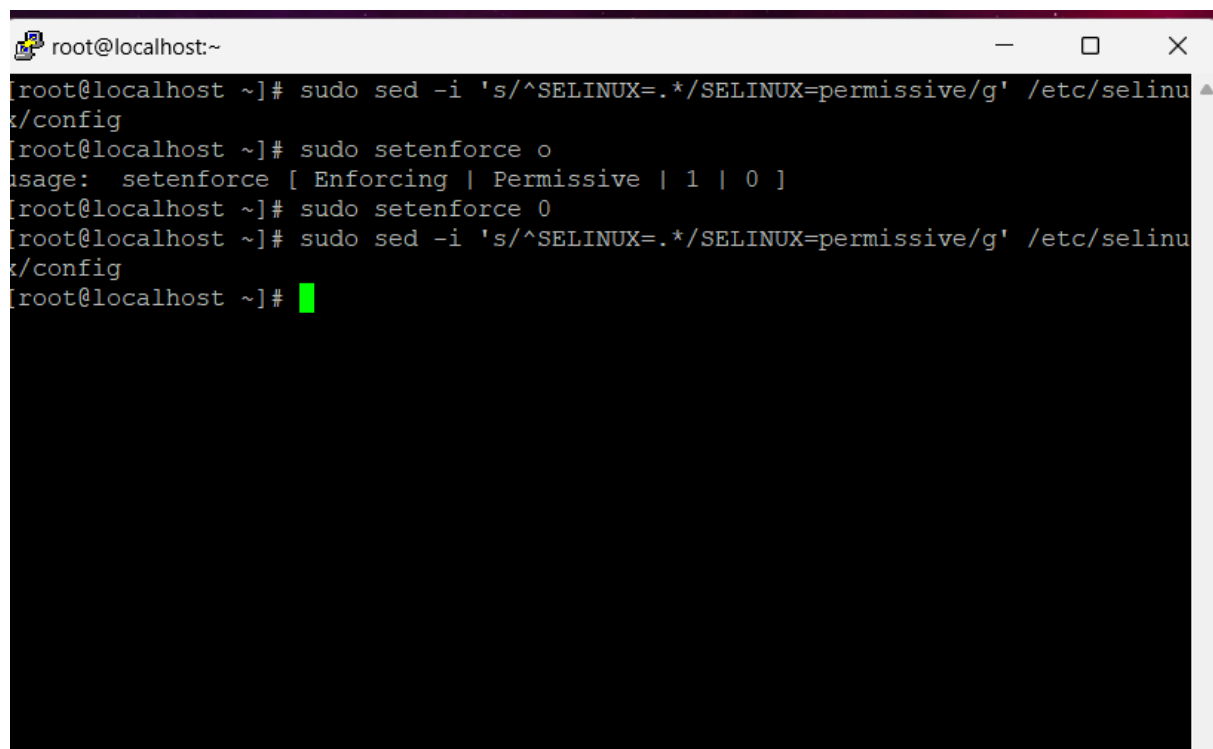
Oct 24 16:09:06 localhost.localdomain systemd[1]: Starting Socket activation for snapd.
Oct 24 16:09:06 localhost.localdomain systemd[1]: Listening on Socket activation for snapd.

root@localhost ~]#

```

Figura 9: Iniciamos y habilitamos el servicio

7. Configuramos SE linux para modo permisivo



```

root@localhost ~]# sudo sed -i 's/^SELINUX=.*/SELINUX=permissive/g' /etc/selinux/config
root@localhost ~]# sudo setenforce 0
Usage:  setenforce [ Enforcing | Permissive | 1 | 0 ]
root@localhost ~]# sudo setenforce 0
root@localhost ~]# sudo sed -i 's/^SELINUX=.*/SELINUX=permissive/g' /etc/selinux/config
root@localhost ~]#

```

Figura 10: modo permisivo

8. Instalamos Mikro8s utilizando el siguiente comando

```
[root@localhost ~]# sudo snap install microk8s --classic
Download snap "snapd" (20290) from channel "stable" 35% 1.43MB/s 19.5s
```

Figura 11: Instalamos Mikro8s

9. Configurar los siguientes permisos para que no exista errores

```
[root@localhost ~]# sudo usermod -a -G microk8s $USER
[root@localhost ~]# sudo chown -f -R $USER ~/.kube
[root@localhost ~]#
```

Figura 12: permisos

10. Ejecutamos algunos comandos newgrp microk8s init 6 microk8s status

```
root@localhost:~
login as: prueba
prueba@192.168.149.131's password:
Last login: Tue Oct 24 13:20:16 2023 from 192.168.149.1
-bash: 'export: command not found
[prueba@localhost ~]$ su -
Password:
Last login: Tue Oct 24 16:37:42 -05 2023 on tty1
-bash: 'export: command not found
[root@localhost ~]# microk8s status
microk8s is running
high-availability: no
  datastore master nodes: 127.0.0.1:19001
  datastore standby nodes: none
addons:
  enabled:
    dns                # (core) CoreDNS
    ha-cluster          # (core) Configure high availability on the current node
    helm                # (core) Helm - the package manager for Kubernetes
    helm3               # (core) Helm 3 - the package manager for Kubernetes
  disabled:
    cert-manager        # (core) Cloud native certificate management
    community            # (core) The community addons repository
    dashboard           # (core) The Kubernetes dashboard
```

Figura 13: Ingresamos los comandos

11. Verificamos los nodos disponibles


```
[root@localhost ~]# microk8s kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
localhost.localdomain Ready    <none>   19m   v1.27.6
[root@localhost ~]#
```

Figura 14: Nodos Disponibles

12. Comprobamos la versión de kubectl

```
[root@localhost ~]# microk8s kubectl
kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/

Basic Commands (Beginner):
  create      Create a resource from a file or from stdin
  expose      Take a replication controller, service, deployment or pod and
expose it as a new Kubernetes service
  run         Run a particular image on the cluster
  set         Set specific features on objects

Basic Commands (Intermediate):
  explain     Get documentation for a resource
```

Figura 15: versión de kubectl

13. Obtenemos las configuraciones debidas para las sentencias

```
QVBCZ05WSFJNQkFmOEVCVEFEQVFIL01BMEdDU3FHU01lM0RRRUJDD1VBQTRJQgpBUUJCRCkZRBkZlZlU
YlJSRnh0RFZzMnRnMUclbHNLb2VsdWxqZThVZmZXdWhtNjZCVW9zNmJialVhNDFPenhjCitmaDBhLzhW
b0dKWmtpMlRSblBTaUFkSWlyMXROcWpSQWc2bzBZOUtMQ0NYQXJOem5ibUJQZ21FK2E0alFIK0wKK2V0
emRBTTRMQkhbk1LamJiMm54OXJaTXVoVlZCc0hySGxMWU5HdExzQ2VsQzJBV0NBRFd6Z2podWtLOEpY
LwpLSktkUG5WcjA4d0VddVo0Z0NIeCtxZzZBUXQzNVI4MlB4TzVSTUVLMFIwZEx2cElBYnp4azh5SWpL
RXBMZyZkClVlaE95MDRrWC9JNDBEaDh2UHZ0eWpYVERBL0lMYzRps1hMTTY2L2VRRkVYeVF0akpZQkpD
SUPPaFFIenVpr1cKdU5Pa2dlOHFMVWpiROVZSWpFUWJhVWtVci0tLS0tRU5EIEENFULRJRklDQVRFLS0t
LS0K
  server: https://192.168.149.131:6443
  name: microk8s-cluster
contexts:
- context:
  cluster: microk8s-cluster
  user: admin
  name: microk8s
current-context: microk8s
kind: Config
preferences: {}
users:
- name: admin
  user:
    token: a0wxMmVsMzBiNi9WWM5FdjNxVTZLRzI4dVcxYU9RUUo4ZVpkU2hNZUU4cz0K

[root@localhost ~]#
```

Figura 16: configuraciones debidas para las sentencias

14. Instalamos kubectl en Rocky Linux 9/AlmaLinux 9 usando el comando:

```
[root@localhost ~]# curl -LO https://storage.googleapis.com/kubernetes-release/r
elease/v1.21.2/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
 33 44.2M   33 14.7M    0     0  398k      0  0:01:53  0:00:37  0:01:16 85885
```

Figura 17: Instalamos kubectl en Rocky Linux 9/AlmaLinux 9

15. Damos los permisos

```
[root@localhost ~]# sudo chmod +x kubectl
[root@localhost ~]#
```

Figura 18: Damos los permisos

16. Lo movemos a donde lo necesitamos

```
root@localhost ~]# sudo mv kubectl /usr/local/bin/
root@localhost ~]#
```

Figura 19: Lo movemos a donde lo necesitamos

17. Realizamos la configuración necesaria

```
[root@localhost ~]# cd $HOME
[root@localhost ~]# microk8s config > ~/.kube/config
[root@localhost ~]#
```

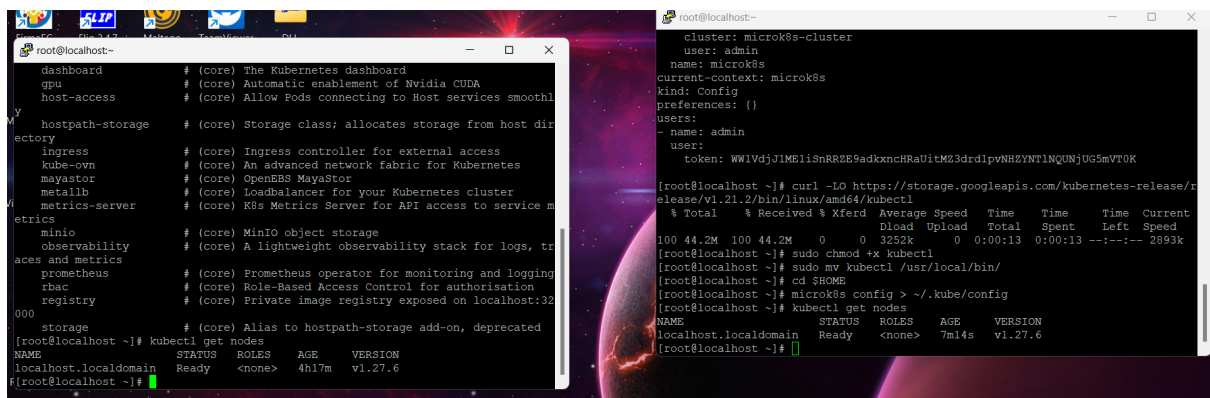
Figura 20: configuración necesaria

18. Verificamos los nodos disponibles

```
[root@localhost ~]# kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
localhost.localdomain Ready    <none>   57m   v1.27.6
[root@localhost ~]#
```

Figura 21: nodos disponibles

19. Necesitamos tener otra maquina virtual que actue en modo esclavo



```
root@localhost~
dashboard # (core) The Kubernetes dashboard
gpu # (core) Automatic enablement of Nvidia CUDA
host-access # (core) Allow Pods connecting to Host services smoothly
hostpath-storage # (core) Storage class; allocates storage from host disk
ingress # (core) Ingress controller for external access
kube-ovn # (core) An advanced network fabric for Kubernetes
mayastor # (core) OpenEBS MayaStor
metallb # (core) LoadBalancer for your Kubernetes cluster
metrics-server # (core) K8s Metrics Server for API access to service metrics
minio # (core) MinIO object storage
observability # (core) A lightweight observability stack for logs, traces and metrics
prometheus # (core) Prometheus operator for monitoring and logging
rbac # (core) Role-Based Access Control for authorisation
registry # (core) Private image registry exposed on localhost:32000
storage # (core) Alias to hostpath-storage add-on, deprecated

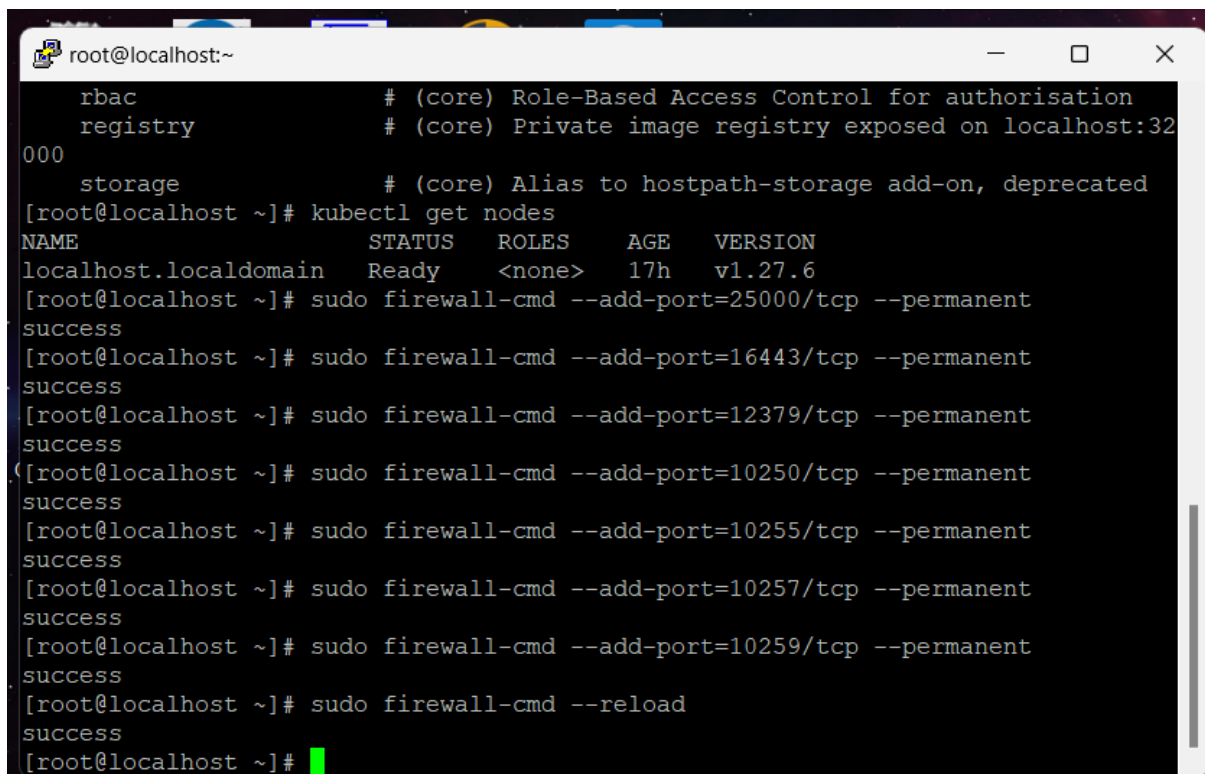
[root@localhost ~]# kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
localhost.localdomain Ready    <none>   4h17m   v1.27.6
[root@localhost ~]#
```

```
cluster: microk8s-cluster
user: admin
name: microk8s
current-context: microk8s
kind: Config
preferences: {}
users:
- name: admin
  user:
    token: WW1VdjJlMEliSnRRZE9adKxncHRAUitM23drdlpvNHZyNTlNQUNjUG5mVTOK

[root@localhost ~]# curl -LO https://storage.googleapis.com/kubernetes-release/r
elease/v1.21.2/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 44.2M  100 44.2M    0     0 3252k      0  0:00:13  0:00:13  ---:-- 2893k
[root@localhost ~]# sudo chmod +x kubectl
[root@localhost ~]# sudo mv kubectl /usr/local/bin/
[root@localhost ~]# cd $HOME
[root@localhost ~]# microk8s config > ~/.kube/config
[root@localhost ~]# kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
localhost.localdomain Ready    <none>   7m14s   v1.27.6
[root@localhost ~]#
```

Figura 22: otra maquina virtual

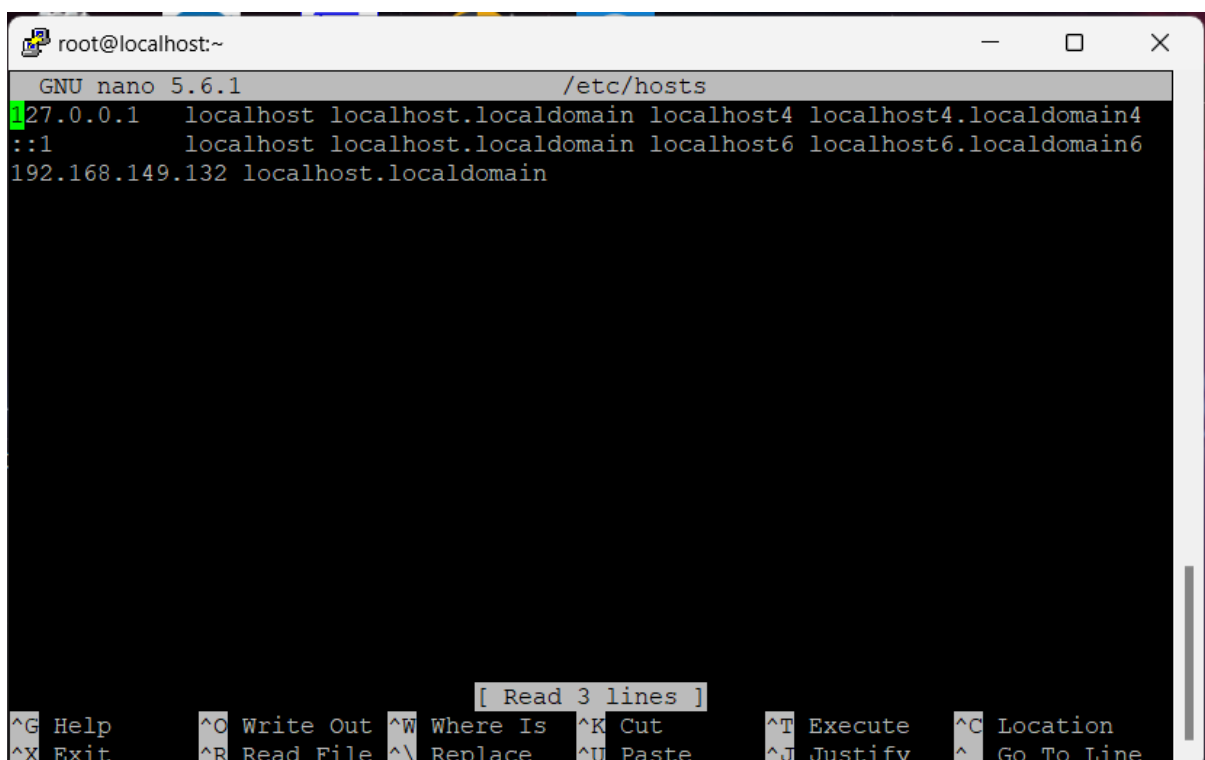
20. En el nodo MASTER agregue la siguiente configuración para los puertos



```
root@localhost:~  
rbac # (core) Role-Based Access Control for authorisation  
registry # (core) Private image registry exposed on localhost:32000  
storage # (core) Alias to hostpath-storage add-on, deprecated  
[root@localhost ~]# kubectl get nodes  
NAME STATUS ROLES AGE VERSION  
localhost.localdomain Ready <none> 17h v1.27.6  
[root@localhost ~]# sudo firewall-cmd --add-port=25000/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=16443/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=12379/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=10250/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=10255/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=10257/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=10259/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --reload  
success  
[root@localhost ~]#
```

Figura 23: nodo MASTER

21. Añadir en la direccion /etc/hosts la direccion ip del nodo esclavo



```
GNU nano 5.6.1 /etc/hosts  
27.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  
192.168.149.132 localhost.localdomain  
[ Read 3 lines ]  
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^_ Go To Line
```

Figura 24: direccion ip del nodo esclavo

22. Se genera un comando que utilizaran los nodos para unirse al nodo master docker-compose up -d

```
root@localhost:~  
microk8s join 172.19.0.1:25000/a1d70c440eb66aac9becdab309a27fa5/354f3544d5d8  
[root@localhost ~]# ^C  
[root@localhost ~]# nano /etc/hosts  
[root@localhost ~]# microk8s add-node  
From the node you wish to join to this cluster, run the following:  
microk8s join 192.168.149.131:25000/4fe14db09c5ef073febfedf20308de0c/354f3544d5d8  
3  
Use the '--worker' flag to join a node as a worker not running the control plane  
eg:  
microk8s join 192.168.149.131:25000/4fe14db09c5ef073febfedf20308de0c/354f3544d5d8  
3 --worker  
If the node you are adding is not reachable through the default interface you can  
use one of the following:  
microk8s join 192.168.149.131:25000/4fe14db09c5ef073febfedf20308de0c/354f3544d5d8  
3  
microk8s join 192.168.1.80:25000/4fe14db09c5ef073febfedf20308de0c/354f3544d5d8  
microk8s join 172.17.0.1:25000/4fe14db09c5ef073febfedf20308de0c/354f3544d5d8  
microk8s join 172.20.0.1:25000/4fe14db09c5ef073febfedf20308de0c/354f3544d5d8  
microk8s join 172.18.0.1:25000/4fe14db09c5ef073febfedf20308de0c/354f3544d5d8  
microk8s join 172.19.0.1:25000/4fe14db09c5ef073febfedf20308de0c/354f3544d5d8  
[root@localhost ~]# ^C  
[root@localhost ~]#
```

Figura 25: comando que utilizaran los nodos para unirse al nodo master

23. Ahora en el NODO ESCLAVO digitamos el siguiente código

```
[root@localhost ~]# export OPENSSL_CONF=/var/lib/snapd/snap/microk8s/current/etc  
/ssl/openssl.cnf  
[root@localhost ~]#
```

Figura 26: digitamos el siguiente código

24. De la misma manera con los puertos que vamos a necesitar

```
root@localhost:~  
declare -x SHLVL="1"  
declare -x TERM="xterm"  
declare -x USER="root"  
declare -x XDG_DATA_DIRS="/usr/local/share:/usr/share:/var/lib/snapd/desktop"  
declare -x which_declare="declare -f"  
[root@localhost ~]# export OPENSSL_CONF=/var/lib/snapd/snap/microk8s/current/etc  
/ssl/openssl.cnf  
[root@localhost ~]# sudo firewall-cmd --add-port=25000/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=16443/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=12379/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=10250/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=10255/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=10257/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --add-port=10259/tcp --permanent  
success  
[root@localhost ~]# sudo firewall-cmd --reload  
success  
[root@localhost ~]#
```

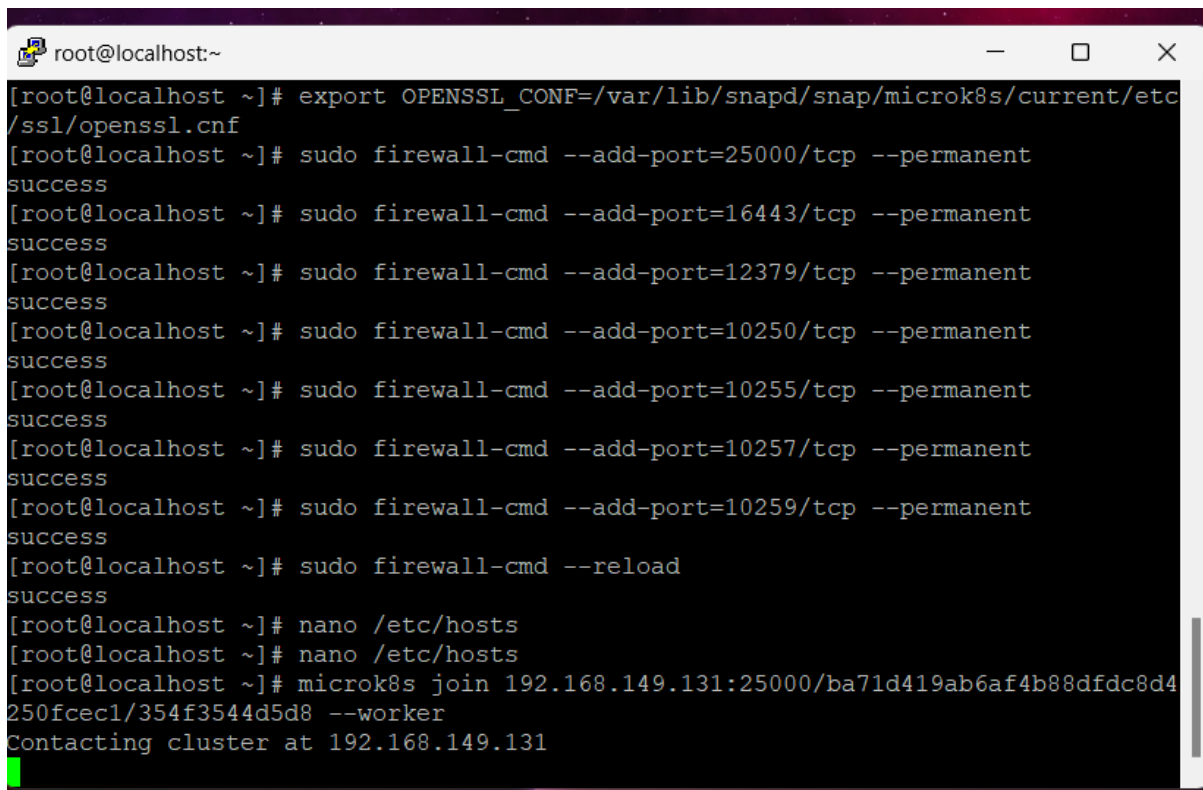
Figura 27: puertos que vamos a necesitar

25. Añadimos la dirección a nuestro archivo etc/hosts de nuestro nodo esclavo

```
GNU nano 5.6.1 /etc/hosts  
27.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
:1 localhost localhost.localdomain localhost6 localhost6.localdomain6  
92.168.149.132 localhost.localdomain  
[ Read 3 lines ]  
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^_ Go To Line
```

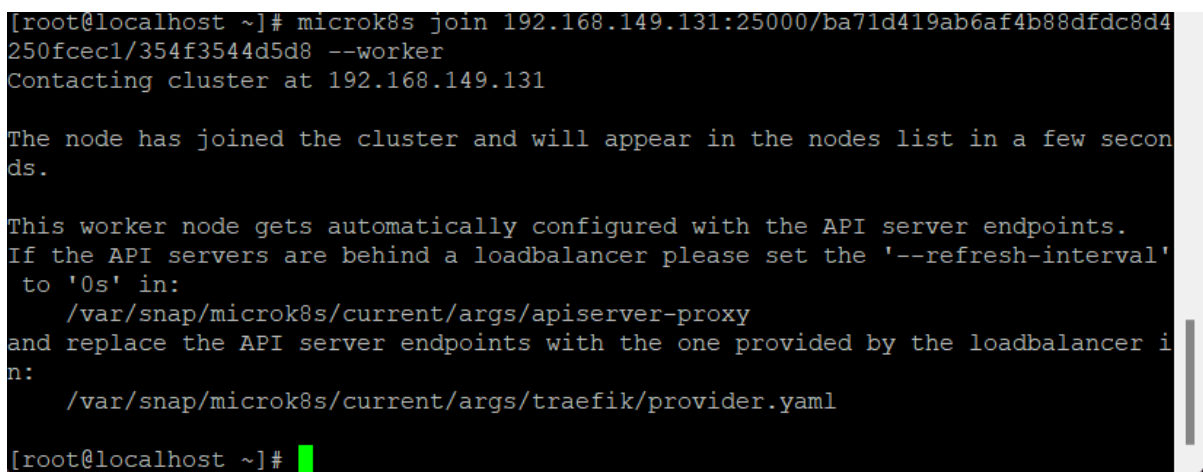
Figura 28: Añadimos la dirección

26. Colocamos el comando generado en nuestro NODO MAESTRO para que se una

A terminal window with a dark background and light text. The prompt is [root@localhost ~]. The user enters several commands: export OPENSSL_CONF=/var/lib/snapd/snap/microk8s/current/etc/ssl/openssl.cnf, followed by seven instances of sudo firewall-cmd --add-port=25000/tcp --permanent, then sudo firewall-cmd --add-port=16443/tcp --permanent, then sudo firewall-cmd --add-port=12379/tcp --permanent, then sudo firewall-cmd --add-port=10250/tcp --permanent, then sudo firewall-cmd --add-port=10255/tcp --permanent, then sudo firewall-cmd --add-port=10257/tcp --permanent, then sudo firewall-cmd --add-port=10259/tcp --permanent, then sudo firewall-cmd --reload, then nano /etc/hosts, then nano /etc/hosts, then microk8s join 192.168.149.131:25000/ba71d419ab6af4b88dfdc8d4250fcec1/354f3544d5d8 --worker. The output shows success for each firewall command and the start of the microk8s join process, including 'Contacting cluster at 192.168.149.131'.

```
[root@localhost ~]# export OPENSSL_CONF=/var/lib/snapd/snap/microk8s/current/etc/ssl/openssl.cnf
[root@localhost ~]# sudo firewall-cmd --add-port=25000/tcp --permanent
success
[root@localhost ~]# sudo firewall-cmd --add-port=16443/tcp --permanent
success
[root@localhost ~]# sudo firewall-cmd --add-port=12379/tcp --permanent
success
[root@localhost ~]# sudo firewall-cmd --add-port=10250/tcp --permanent
success
[root@localhost ~]# sudo firewall-cmd --add-port=10255/tcp --permanent
success
[root@localhost ~]# sudo firewall-cmd --add-port=10257/tcp --permanent
success
[root@localhost ~]# sudo firewall-cmd --add-port=10259/tcp --permanent
success
[root@localhost ~]# sudo firewall-cmd --reload
success
[root@localhost ~]# nano /etc/hosts
[root@localhost ~]# nano /etc/hosts
[root@localhost ~]# microk8s join 192.168.149.131:25000/ba71d419ab6af4b88dfdc8d4250fcec1/354f3544d5d8 --worker
Contacting cluster at 192.168.149.131
```

Figura 29: comando generado

A terminal window showing the output of the microk8s join command. The prompt is [root@localhost ~]. The user enters microk8s join 192.168.149.131:25000/ba71d419ab6af4b88dfdc8d4250fcec1/354f3544d5d8 --worker. The output shows 'Contacting cluster at 192.168.149.131', followed by 'The node has joined the cluster and will appear in the nodes list in a few seconds.' and 'This worker node gets automatically configured with the API server endpoints. If the API servers are behind a loadbalancer please set the '--refresh-interval' to '0s' in: /var/snap/microk8s/current/args/apiserver-proxy and replace the API server endpoints with the one provided by the loadbalancer in: /var/snap/microk8s/current/args/traefik/provider.yaml'. The prompt returns to [root@localhost ~].

```
[root@localhost ~]# microk8s join 192.168.149.131:25000/ba71d419ab6af4b88dfdc8d4250fcec1/354f3544d5d8 --worker
Contacting cluster at 192.168.149.131

The node has joined the cluster and will appear in the nodes list in a few seconds.

This worker node gets automatically configured with the API server endpoints.
If the API servers are behind a loadbalancer please set the '--refresh-interval'
to '0s' in:
    /var/snap/microk8s/current/args/apiserver-proxy
and replace the API server endpoints with the one provided by the loadbalancer in:
    /var/snap/microk8s/current/args/traefik/provider.yaml

[root@localhost ~]#
```

Figura 30: comando generado

5. CONCLUSIONES

- Se realizo la descarga e instalacion de algunos comandos necesarios para el funcionamiento de microk8s
- Se permite la activacion de modo permanente de algunos puertos de firewall necesarios para el enlace.
- Al realizar la configuracion tanto del nodo master como esclavo se ha generado en el nodo master un microk8s join que nos permite enlazar al nodo master.

6. BIBLIOGRAFÍA

[1]<https://voidnull.es/instalacion-de-docker-en-almalinux-9-1/>
<https://www.redhat.com/es/topics/containers/what-is-kubernetes>
<https://kubernetes.io/es/docs/concepts/architecture/nodes/>

7. ANEXOS

```
INSTALCION DE KUBERNETES
sudo dnf update
sudo dnf install epel-release

sudo dnf install snapd

sudo ln -s /var/lib/snapd/snap /snap

echo 'export PATH=$PATH:/var/lib/snapd/snap/bin' | sudo tee -a /etc/profile.d/snap.sh source /etc

sudo systemctl enable --now snapd.socket

systemctl status snapd.socket

sudo setenforce 0
sudo sed -i 's/^SELINUX=.*SELINUX=permissive/g' /etc/selinux/config

sudo snap install microk8s --classic

sudo usermod -a -G microk8s $USER
sudo chown -f -R $USER ~/.kube

newgrp microk8s

init 6

microk8s status

microk8s kubectl get nodes

microk8s config

curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.21.2/bin/linux/amd64/kubectl

sudo chmod +x kubectl
```

Figura 31: ANEXO 1

```

sudo mv kubect1 /usr/local/bin/

cd $HOME
microk8s config > ~/.kube/config

kubect1 get nodes

~~~~~
configuracion del nodo master
~~~~~

Incorporacion de nodos al clúster Microk8s
Requisitos:
Maquina 1 Alma Linux 9: Nodo Master
Maquina 2 Alma Linux 9: Nodo esclavo
En todas las máquinas se debe tener instalado hasta el paso anterior.
En el nodo master, permita que los puertos requeridos pasen a traves del firewall:
sudo firewall-cmd --add-port=25000/tcp --permanent
sudo firewall-cmd --add-port=16443/tcp --permanent
sudo firewall-cmd --add-port=12379/tcp --permanent
sudo firewall-cmd --add-port=10250/tcp --permanent
sudo firewall-cmd --add-port=10255/tcp --permanent
sudo firewall-cmd --add-port=10257/tcp --permanent
sudo firewall-cmd --add-port=10259/tcp --permanent

sudo firewall-cmd --reload

*Añada en la direccion /etc/hosts la direccion ip del nodo esclavo

nano /etc/hosts

192.168.149.132 localhost.localdomain

*Genere el comando que utilizaran los nodos para unirse al cluster

microk8s add-node

```

Figura 32: ANEXO 2


```

microk8s add-node

microk8s join 192.168.149.131:25000/677a4ed2fab6f6c33b3c703a93ebd228/354f3544d5d8 --worker

////////////////////////////////////*****
configuracion del nodo esclavo
////////////////////////////////////*****

*Ahora en el nodo esclavo (Máquina 2) después de instalar Microk8s en el nodo esclavo, configuramos
lo siguiente:

export OPENSSL_CONF=/var/lib/snapd/snap/microk8s/current/etc/ssl/openssl.cnf
sudo firewall-cmd --add-port=25000/tcp --permanent
sudo firewall-cmd --add-port=16443/tcp --permanent
sudo firewall-cmd --add-port=12379/tcp --permanent
sudo firewall-cmd --add-port=10250/tcp --permanent
sudo firewall-cmd --add-port=10255/tcp --permanent
sudo firewall-cmd --add-port=10257/tcp --permanent
sudo firewall-cmd --add-port=10259/tcp --permanent

sudo firewall-cmd --reload

Añada en la direccion /etc/hosts la direccion ip del nodo esclavo

nano /etc/hosts

192.168.149.131 localhost.localdomain

Ahora use el comando generado en el maestro para unir los nodos al clúster Microk8s.

microk8s join 192.168.149.131:25000/677a4ed2fab6f6c33b3c703a93ebd228/354f3544d5d8 --worker

*Para comprobar
kubectl get nodes

*para abandonar un nodo maestro
microk8s leave

```

Figura 33: ANEXO 3