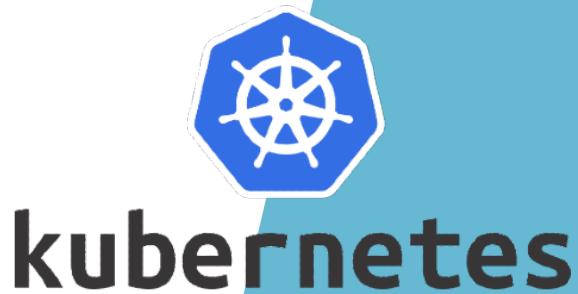


TFC – Pedro José Moldenhauer López

Despliegue de catálogo de aplicaciones
sobre Kubernetes e Hipervconvergencia
basada en Harvester y Rancher



Indice

Indice.....	2
Estudio del problema y análisis del sistema.....	4
Introducción.....	4
Funciones y rendimientos deseados.....	5
Objetivos.....	5
Planteamiento y evaluación de diversas soluciones y justificación de la solución elegida.....	5
Modelado de la solución.....	6
Recursos humanos.....	6
Recursos hardware.....	6
Recursos software.....	6
Planificación temporal. Etapas.....	7
Ejecución del proyecto y elaboración de la documentación técnica.....	8
Infraestructura del proyecto.....	8
Network.....	9
Fase de pruebas con la creación de una batería de pruebas.....	11
Documentación del sistema.....	12
Introducción a la aplicación.....	12
Interfaz de clúster de Rancher.....	12
Interfaz de Harvester.....	13
Interfaz de ArgoCD.....	13
Manual de Instalación.....	14
Instalación de Harvester.....	14
Requisitos.....	14
Hardware utilizado.....	14
Instalación.....	15
Instalación de Rancher.....	18
Requisitos.....	18
Sistema operativo.....	19
Configuración básica del firewall.....	19
Configuración clave ssh.....	20
Instalación.....	20
Instalación de ArgoCD.....	26
Instalación de Haproxy.....	29
Archivo de configuración de Haproxy.....	29
Despliegue de Haproxy.....	30
Manuales de administración.....	34
Conectar el clúster de Harvester con Rancher.....	34
Configuración de las redes en Harvester.....	38
Creación de las redes.....	38
Asignar la red de storage.....	41
Asignar la red para las máquinas virtuales.....	42
Creación del cluster de Kubernetes en Rancher sobre el clúster de Harvester.....	43
Preparación de la imagen para los nodos.....	43
Creación de máquina virtual en Harvester (Prueba de la imagen).....	44
Creación del clúster.....	47
Administrar el nuevo clúster.....	56
Manuales de Usuario.....	58
Despliegue de aplicación en ArgoCD.....	58
Configurar Ingress.....	62
Cluster local.....	62
Cluster ArgoCD.....	65
Certificado Letsencrypt.....	67
Conclusiones finales.....	68
Grado de cumplimiento de los objetivos fijados.....	69
Bibliografía empleada.....	70
Instalacion-Configuración Harvester.....	70
Instalacion Rancher.....	70
Cluster Kubernetes.....	70
ArgoCD.....	71
HaProxy.....	71
Ansible.....	71
Letsencrypt.....	71

Estudio del problema y análisis del sistema.

- El problema/necesidad al que intento dar una solución con este proyecto seria la de ofrecer una manera de desplegar aplicaciones de forma segura, con actualización automática y con un fácil escalado o des-escalado. Además, todo esto debe ser en un entorno con alta disponibilidad para que las aplicaciones/servicios estén siempre disponibles pase lo que pase y altamente personalizable para poder adaptarlo a cualquier necesidad de cualquier entorno o aplicación.

Como es tendencia hoy en día, la mejor forma en la que creo que se puede ofrecer este servicio seria a través de una estructura de micro-servicios contenidos en contenedores de docker y gestionados con kubernetes, aunque siempre teniendo en cuenta que a veces es mejor descartar la opción de los mircro-servicios y usar máquinas completas para ciertos servicios o aplicaciones, por eso, mi proyecto ofrece todas esas capacidades.

Introducción

- El proyecto consiste en el despliegue de una infraestructura de virtualización basada en kubernetes y kvm a través de un software llamado Harvester.

El software de Harvester es un hipervisor opensource de última generación que permite crear tanto clústeres de kubernetes como máquinas virtuales con kvm de manera sencilla y eficiente, este software esta pensado para ser utilizado en el entorno empresarial y dentro de los hipervisores opensource es de los más modernos actualmente. La infraestructura consiste en tres servidores físicos en los que se ha instalado Harvester y se ha formado un clúster con los tres, este clúster se gestiona a través del gestor Rancher, el cual se aloja en una máquina virtual separada de la infraestructura del clúster y, a parte de permitir gestionar el clúster de Harvester, permite gestionar los clusteres de kubernetes creados dentro de harvester.

Dentro del clúster de kubernetes se podrán desplegar y actualizar aplicaciones a través de una herramienta GitOps llamada ArgoCD, esta herramienta permite conectar repositorios de Helm o Github y desplegarlos automáticamente en el clúster de kubernetes, también permite automatizar las actualizaciones de la aplicación que hemos instalado cada vez que se actualice el repositorio. Además, permite la creación de distintos usuarios con distintos permisos de acceso a los recursos y aplicaciones desplegadas y ofrece información sobre el estado de las aplicaciones y sus repositorios.

Todas las interfaces que estén expuestas a el exterior están securizadas con sus respectivos certificados y firewalls y estarán asociadas a un nombre dns para su acceso.

También, al estar basada en clústeres, replicada y tener varios enlaces duplicados a la red, esta infraestructura tiene alta disponibilidad.

Funciones y rendimientos deseados

- Las funciones y rendimientos objetivos de este proyecto podrían ser tanto crear un sistema completo para poner una gran cantidad de aplicaciones en producción para distintos clientes de manera segura y semi-automatizada como un sistema de testeo donde poder probar aplicaciones en desarrollo fácilmente, o mezclar ambos y poder tener en el mismo clúster varios entornos para distintos propósitos, desarrollo, QA, producción, etc.

Objetivos

- El sistema implementado ofrece tanto servicios de máquinas virtuales completas como despliegue de aplicaciones en kubernetes, con la posibilidad de tener varios clusteres de kubernetes separados para distintos propósitos con diferentes capacidades.

Planteamiento y evaluación de diversas soluciones y justificación de la solución elegida.

- Hoy en día hay muchas posibilidades a la hora de ofrecer servicios de aplicaciones, desde sistemas monolíticos con escalado vertical, como por ejemplo máquinas EC2 de amazon AWS, hasta sistemas de micro-servicios montados en clusteres de k8s en la nube. Sin embargo, la gran mayoría de estos servicios son difíciles de instalar/gestionar, necesitando una gran cantidad de técnicos detrás configurando todo o tiene limitaciones en ciertos aspectos, como por ejemplo estar sujetos a la disponibilidad/reglas de otras empresas (AWS).

Además, muchas empresas siguen prefiriendo tener servidores físicos que puedan controlar al 100% ellos mismos y poder ofrecer los servicios de hosteo a sus clientes de manera directa sin ningún intermediario.

Por todo esto, la solución que he elegido me parece la más apropiada para una empresa grande-mediана, ya que puede ofrecer tanto micro-servicios como máquinas completas y se puede escalar de manera horizontal, añadiendo más servidores a un clúster ya creado, como vertical, sustituyendo los servidores por otros mejores, de manera muy sencilla.

Modelado de la solución

Recursos humanos

- Los recursos humanos para montar/gestionar un proyecto de este estilo no serían muy grandes, un solo técnico podría gestionar una gran cantidad de servidores ya que casi todo se gestiona de manera automática y, una vez montado, lo único que hace falta es desplegar las aplicaciones que se quiera.

Recursos hardware

- Para este proyecto se han usado tres servidores dedicados con 128 GB de memoria ram cada uno, cada servidor tiene 10 interfaces de red de ethernet de 1GB de velocidad y una interfaz extra para la ILO del servidor, de estas interfaces se van a usar 9 de ellas en cada uno. Cada servidor tiene 2 discos ssd de 800GB en una raid 1+0 para la instalación del sistema y otros 2 discos ssd de 980GB en otra raid 1+0 para los datos de las máquinas virtuales.

También se va a usar un switch con funcionalidad de VLan y una máquina virtual externa.

Recursos software

El software que se ha usado para el proyecto es el siguiente:

- Harvester instalado de manera nativa en cada servidor
- Rancher instalado sobre un clúster de kubernetes en una máquina virtual externa con un sistema Linux (Ubuntu 22.04).
- Kubectl, instalado en la máquina de rancher y en mi ordenador.
- Helm, instalado en la máquina de rancher.
- RKE, instalado en la máquina de rancher y en el clúster virtual para argocd.
- ArgoCD, instalado en el clúster virtual de kubernetes.
- Longhorn, instalado en el clúster virtual de kubernetes y de manera nativa en Harvester.
- Prometheus y Grafana, instalados en el clúster virtual de kubernetes y de manera nativa en Harvester.
- Haproxy, desplegado en el clúster local de rancher a través de kubectl.

Esos son los principales paquetes software utilizados en el proyecto, aunque hay muchos otros que ya vienen instalados por defecto en los demás paquetes (ingress-nginx, alertmanager, etc.) o que he usado para las aplicaciones de pruebas y no son esenciales para el proyecto (mariadb, php-fpm, etc.)

Planificación temporal. Etapas.

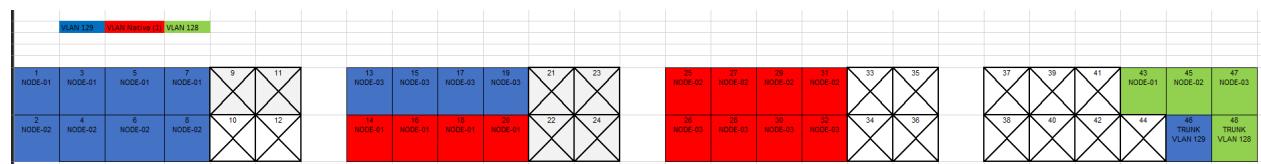
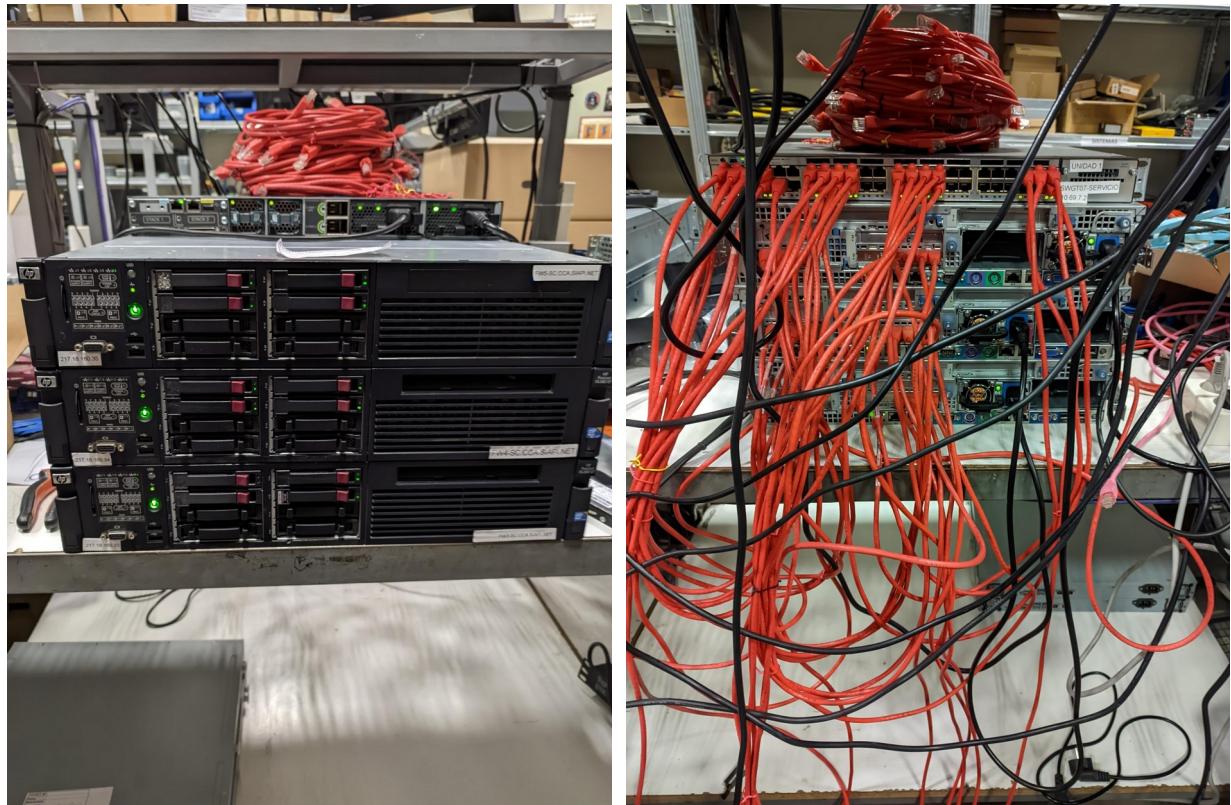
- Estas son las etapas que he seguido a la hora de la realización del proyecto.

1. Preparar la infraestructura física
 - 1.1. Función
 - 1.2. Documentación
2. Instalar Harvester y crear el clúster
 - 2.1. Función
 - 2.2. Documentación
3. Configuración de la network en el clúster
 - 3.1. Función
 - 3.2. Documentación
4. Instalar Rancher y conectarlo con Harvester
 - 4.1. Función
 - 4.2. Documentación
 - 4.3. Automatización con ansible
5. Crear clúster de kubernetes desde rancher y testearlo
 - 5.1. Creación imagen SO
 - 5.2. Creación clúster
 - 5.3. Documentación
6. Instalar ArgoCD y configurarlo
 - 6.1. Función
 - 6.2. Documentación
7. Desplegar alguna aplicación desde GitHub o Helm
 - 7.1. Función
 - 7.2. Documentación
8. Configurar acceso a las aplicaciones desde internet con haproxy/ingress
 - 8.1. Función
 - 8.2. Documentación
9. Crear certificado para las aplicaciones con Letsencrypt
 - 9.1. Función
 - 9.2. Documentación
10. Documento final recopilando toda la información del proyecto
11. Preparar la presentación y una prueba de despliegue
12. (Extra) Crear aplicación web propia y chart de helm para desplegarla

Ejecución del proyecto y elaboración de la documentación técnica

Infraestructura del proyecto

- Como ya he dicho en la sección de recursos hardware, el proyecto cuenta con 3 servidores físicos, un switch y una máquina virtual, a continuación adjunto algunas fotos de los servidores y un esquema de conexiones del switch.



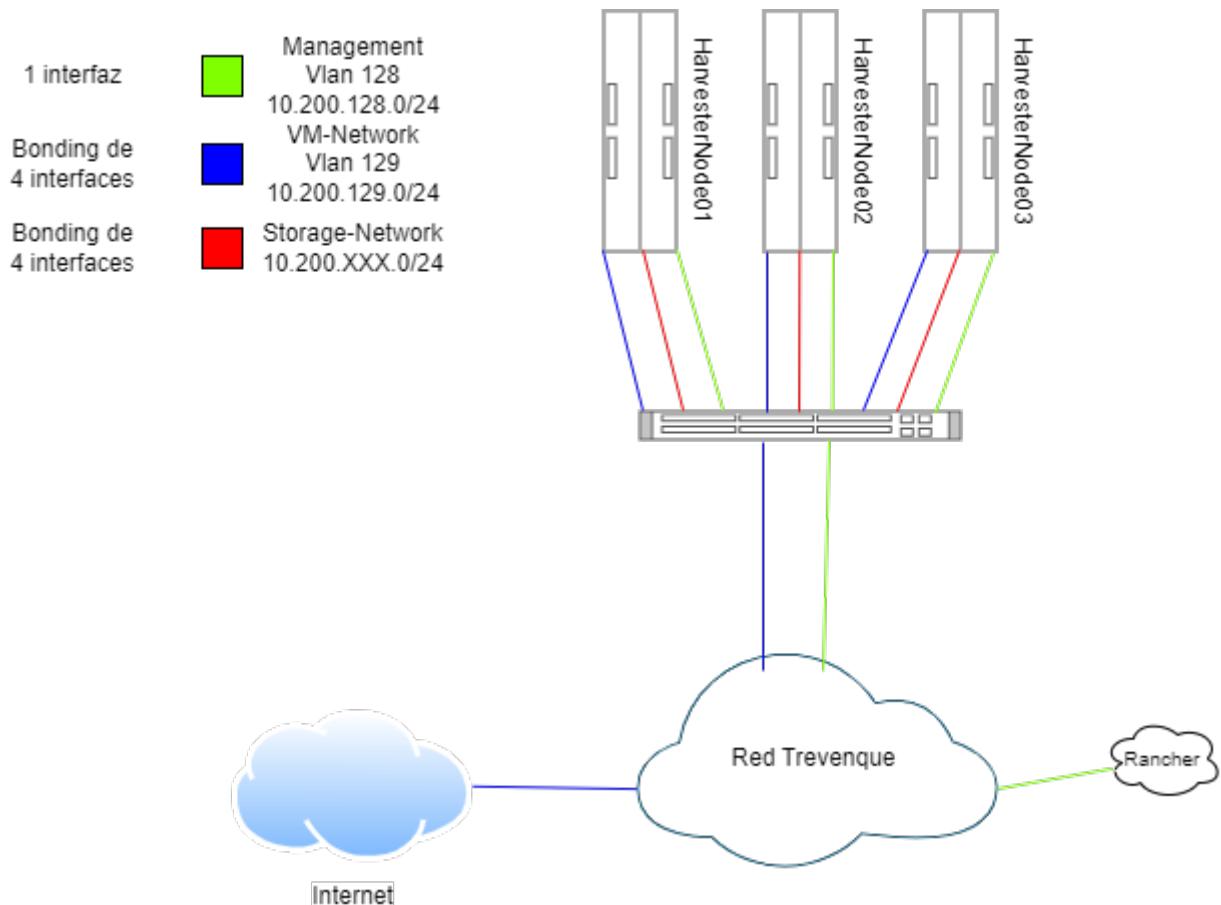
Network

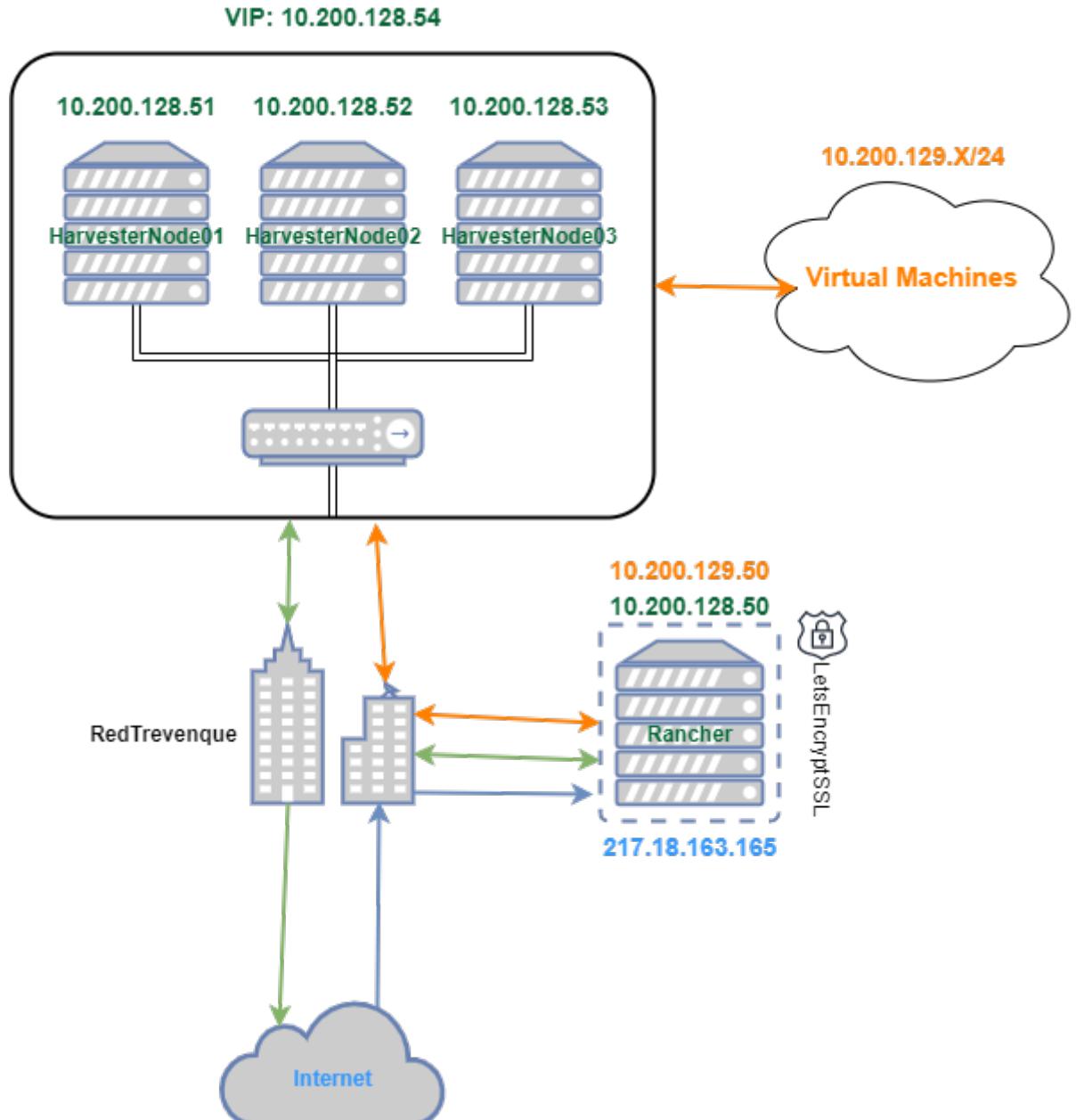
- La red que se ha creado para el proyecto es una estructura de tres subredes privadas, dos de ellas conectadas a la red interna de la empresa a través de Vlan y la tercera sera una red interna del proyecto que conecte solo los servidores entre si.

Las redes van sirven para los siguientes propósitos:

- La subred de **Management** (128) se usa para la gestión tanto de las máquinas virtuales como de los nodos de harvester.
- La subred de **VM** (129) se usa para el acceso a las máquinas virtuales, ya sea a través de la red interna como a internet.
- La subred de **Storage** es una red interna que conecta los servidores del clúster entre si para el almacenamiento compartido de Longhorn (Almacenamiento compartido distribuido).

Para acceder desde fuera a el proyecto se le ha asignado una ip publica a la máquina virtual de rancher, desde la cual se puede acceder a la interfaz web del propio rancher o se reenvia el trafico dirigido a las distintas apps de dentro del clúster de kubernetes.





Fase de pruebas con la creación de una batería de pruebas

- Para probar el proyecto, he desplegado varias aplicaciones en el clúster de kubernetes y he creado distintas máquinas virtuales, con versión de escritorio y de terminal, y he probado el correcto funcionamiento de estas. A continuación, pongo algunas imágenes del resultado final, en la sección de manual de usuario explicaré como he desplegado las aplicaciones y como he creado las máquinas virtuales. A parte de estas dos aplicaciones que muestro en las imágenes, también he desplegado otras dos aplicaciones para mostrarlas durante la presentación del proyecto.

The screenshot shows the Argo UI interface. On the left, there's a sidebar with navigation links for Applications, Settings, User Info, Documentation, Favorites Only, SYNC STATUS, HEALTH STATUS, LABELS, and PROJECTS. The main area displays two application cards:

- guestbook**: Project: default, Labels: none, Status: Healthy Synced, Repository: https://github.com/argoproj/argocd-example-apps, Target Revision: HEAD, Path: helm-guestbook, Destination: in-cluster, Namespace: guestbook, Created At: 05/17/2023 12:52:01 (a minute ago). Buttons: SYNC, REFRESH, DELETE.
- nginx-test**: Project: default, Labels: none, Status: Healthy Synced, Repository: https://charts.bitnami.com/bitnami, Target Revision: 14.2.1, Chart: nginx, Destination: in-cluster, Namespace: nginx-test, Created At: 05/17/2023 12:43:46 (9 minutes ago). Buttons: SYNC, REFRESH, DELETE.

At the top right, there are buttons for APPLICATIONS TILES, Log out, and sorting options (Sort: name, Items per page: 10).

The screenshot shows the nginx welcome page at https://lab2.pmolden.es. The title is "Welcome to nginx!". It includes a message: "If you see this page, the nginx web server is successfully installed and working. Further configuration is required." Below it, it says: "For online documentation and support please refer to [nginx.org](#). Commercial support is available at [nginx.com](#). Thank you for using nginx."

The screenshot shows the Guestbook application at https://lab2.pmolden.es. The title is "Guestbook". There is a search bar with "Query here" and a "Search" button. Below it is a text input field labeled "Messages" with placeholder text "Type your message here...". A "Submit" button is at the bottom left.

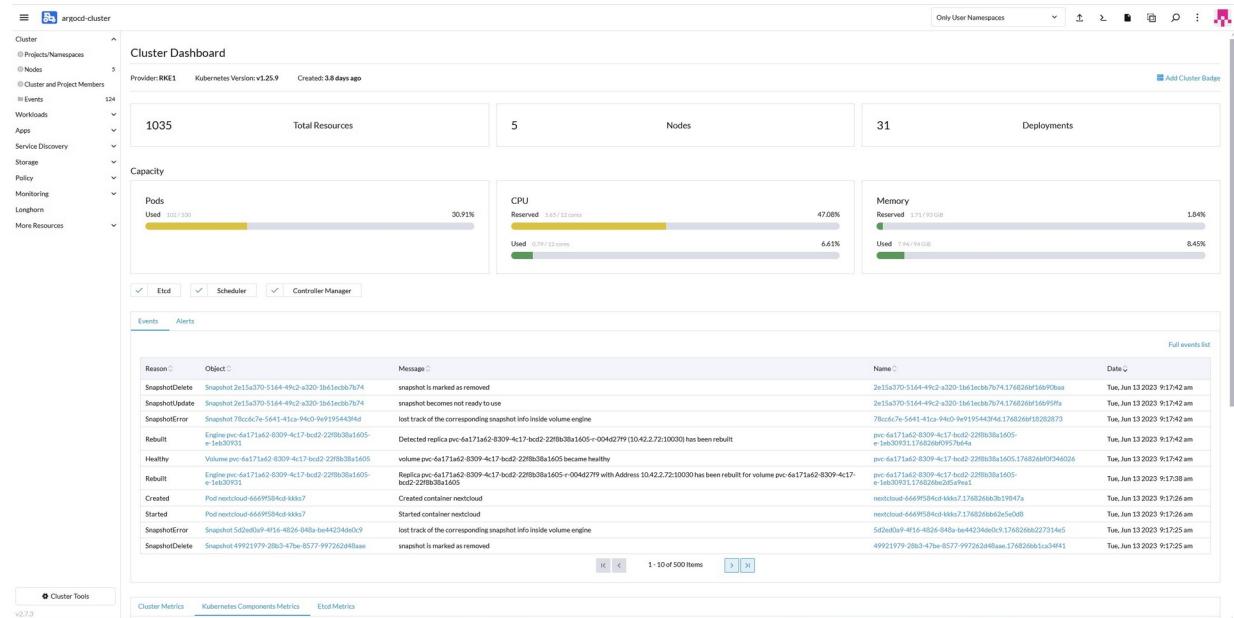
Documentación del sistema

Introducción a la aplicación

- Las distintas aplicaciones/interfaces finales que voy a usar para administrar tanto las maquinas virtuales, como el clúster y las aplicaciones que serán desplegadas en el mismo seran las siguientes:

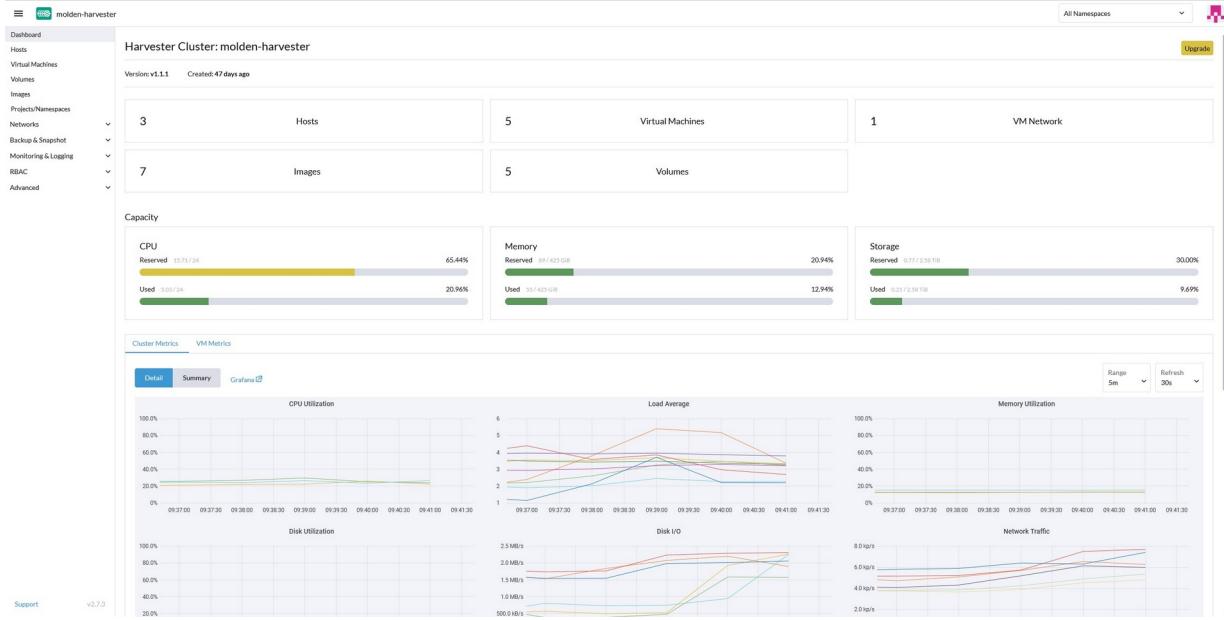
• Interfaz de clúster de Rancher

- En esta interfaz administrare tanto los clusteres como el acceso a las aplicaciones a través de ingress y haproxy, desde esta interfaz también podre acceder a la monitorización de los distintos clusteres de kubernetes y podre acceder a la interfaz de harvester sin tener que estar conectado directamente a la red del mismo.



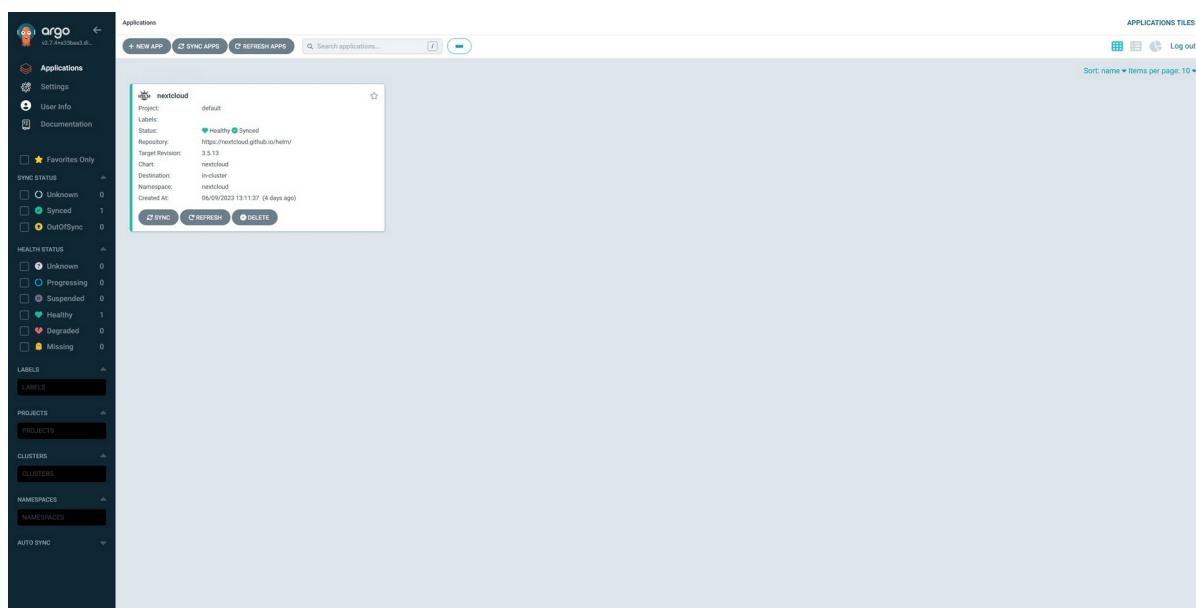
• Interfaz de Harvester

- Desde aquí podre gestionar y monitorizar los nodos físicos del clúster de harvester y crear las maquinas virtuales.



• Interfaz de ArgoCD

- Desde esta interfaz podre gestionar las aplicaciones que tengo instaladas en el clúster de kubernetes y ver su estado.



Manual de Instalación

- Durante el proceso de instalación he ido documentando el proceso para cada uno de los paquetes/aplicaciones que he necesitado para el sistema final, por ello, y para que se pueda leer de una manera más fácil y ordenada, a continuación voy a separar el manual de instalación de cada software instalado.

- **Instalación de Harvester**

Requisitos

- Para poder instalar Harvester en un servidor o en una maquina virtual, debe cumplir unos requisitos mínimos:

- Solo admite CPUs x86_64 compatibles con virtualización y con un mínimo de 8 cores.
- Mínimo 32GB de memoria RAM, aunque en la documentación oficial recomiendan 64GB para un entorno en producción. Con menos de 32GB puede funcionar, pero de manera limitada.
- Mínimo 200GB de capacidad en disco, 500GB recomendados para un entorno en producción. Durante la instalación también recomienda separar en discos distintos la instalación de Harvester de el almacenamiento reservado para las maquinas virtuales.
- Los discos de almacenamiento deben ser SSD con una velocidad mínima de 5000 IOPS.
- Mínimo 1 interfaz ethernet de 1Gbps.

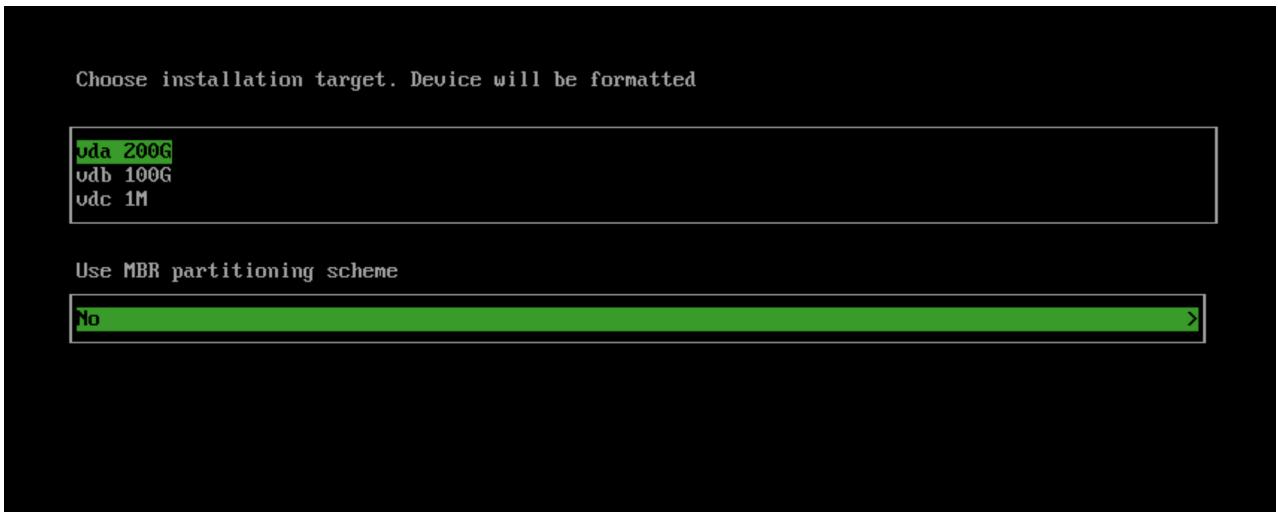
Hardware utilizado

- En el caso de este proyecto, se han usado tres servidores idénticos con las siguientes características:

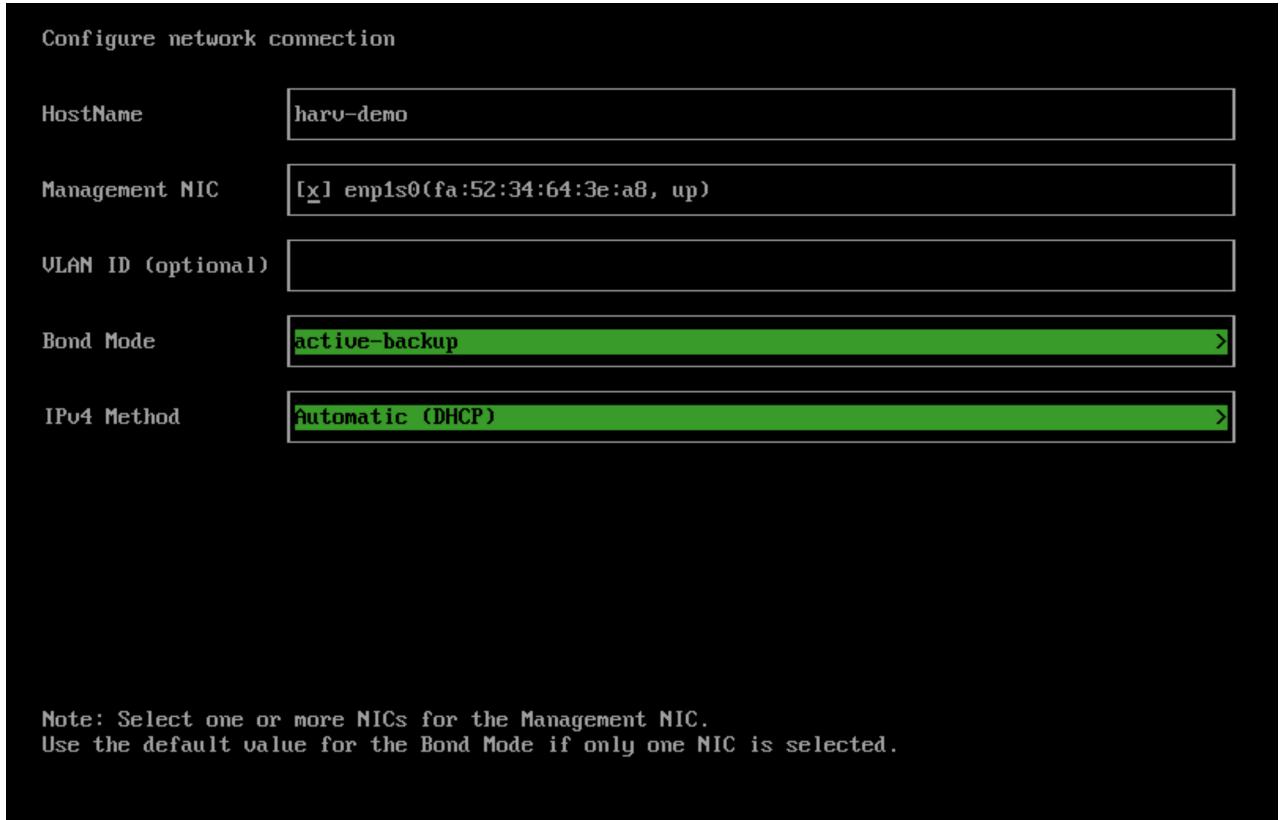
- 128 GB de ram
- 2 CPUs de 3,3ghz
- 2 discos ssd de 800GB en raid 1+0 (Sistema)
- 2 discos ssd de 980GB en raid 1+0 (VM-Data)
- 10 interfaces de red de 1GB utilizadas en cada nodo

Instalación

- Para la instalación del primer nodo hay que seguir los siguientes pasos:
 1. Para instalar Harvester descargo la ISO desde el [repositorio oficial](#) de la versión que queremos instalar, se recomienda siempre usar la última versión de release, actualmente es la 1.1.1 (Esta es la forma de instalación manual, también se podría instalar de manera automática a través de la red con PXE, pero para este proyecto lo haré de forma manual)
 2. Una vez descargado, creo una unidad USB booteable con la ISO usando cualquier programa que sirva para ello, como por ejemplo [Balena Etcher](#) o [Rufus](#)
 3. Conecto la memoria USB a el servidor y enciendo la maquina, selecciono el instalador cuando aparezca el menú y sigo el asistente de instalación. Dentro de este asistente tendré que seleccionar las siguientes opciones:
 - La primera opción sera si quiero crear un nuevo clúster o añadir el nodo a uno existente, como es el primer nodo que instalo creo uno nuevo.
 - Lo siguiente sera elegir en que disco se instalara el sistema y en que disco se guardaran los datos de las maquinas virtuales, según recomiendan en la documentación de harvester, es mejor que sean dos discos distintos, pero se puede usar el mismo. A la hora de elegirlo, el disco del sistema debe tener como mínimo 200GB y el de VM no tiene un mínimo, pero recomienda que sea de 140GB.



- Ahora tendremos que asignarle al nodo un hostname, selecciono la "management nic" que sera la interfaz de red que usare para la gestión del nodo, una vlan id si es necesario, el bond mode y por último en esta pantalla podre elegir de que manera adquirirá una ip en la red de management, puede ser de manera estática, es decir que tendremos que introducir la ip de manera manual o por DHCP.



- En las siguientes pantallas podre configurar un DNS (Opcional), una "Virtual ip (VIP)" la cual es una ip diferente a la del nodo pero en la misma red que harvester usara como ip de acceso al clúster, un token para el clúster el cual se usara más tarde para unir más nodos al clúster, una contraseña para el host (El usuario por defecto es rancher), los servidores ntp que vaya a usar y por último hay tres configuraciones opcionales que son indicar un servidor proxy en caso de ser necesario, importar las claves ssh desde github y si quiero configurar la instalación de harvester de manera automática con un archivo de configuración predefinido.

4. Una vez terminado el asistente de instalación, confirmo la instalación y espero a que aparezca el menú de harvester donde indicara la ip del nodo, la ip del clúster y el estado de cada uno, aquí, si pulso f12, podemos acceder a una terminal. Ya esta instalado harvester, para acceder a la configuración del clúster entro a través de http o https en la dirección ip del clúster. La primera vez que entro harvester pide que configure un usuario y contraseña para el acceso a la interfaz web del clúster.



- Para los nodos siguientes del clúster de harvester:

Una vez instalado el primer nodo y creado el clúster, para añadir un nuevo nodo solo tengo que seguir las instrucciones anteriores, pero en vez de elegir en la primera opción crear un clúster nuevo, elijo añadir a un clúster existente. Todo el resto del proceso sera igual, excepto porque aquí no pedirá un VIP nuevo, si no el ya existente del clúster y también pedirá el token que escribí anteriormente.

Una vez he acabado todo el proceso de instalación para configurar el clúster lo puedo hacer desde la interfaz web.

• Instalación de Rancher

- A parte de la instalación manual de rancher, he preparado un playbook de ansible que te instala de manera automática todo, desde el cluster de RKE hasta rancher. Como copiarlo en este documento seria demasiado código, dejo el enlace de github del proyecto donde esta. <https://github.com/firex20/ProyectoFinalCiclo/tree/main/Ansible/Rancher>

Requisitos

- Para instalar Rancher debo tener un entorno con Kubernetes instalado, ya sea en un solo nodo o en varios. También tengo otras opciones de instalación, como por ejemplo una instalación automática desde EKS de amazon o una instalación dentro de un contenedor de docker.

En este proyecto voy a usar una única maquina virtual fuera del clúster de Harvester, pero dentro de la red de la empresa, ya que los servidores no van a tener acceso directo desde internet con ip publica, si no que sera la propia maquina de Rancher a la que se podrá acceder desde fuera. Tampoco voy a usar el contenedor de Docker ya que esta recomendado solo para entornos de prueba.

Los requisitos de la maquina virtual que se necesitan varían según la cantidad de clúster y nodos que vaya a manejar con esta instancia de Rancher, también depende de el tipo de instalación que se vaya a realizar, en este caso, como voy a hacer un clúster de un solo nodo de kubernetes para alojar el servidor de Rancher, los requisitos que se aplican son los siguientes:

Deployment Size	Clusters	Nodes	vCPUs	RAM
Small	Up to 150	Up to 1500	2	8 GB
Medium	Up to 300	Up to 3000	4	16 GB
Large	Up to 500	Up to 5000	8	32 GB
X-Large	Up to 1000	Up to 10,000	16	64 GB
XX-Large	Up to 2000	Up to 20,000	32	128 GB

En el caso de este proyecto, ya que no voy a manejar una gran cantidad de nodos ni de clusters, usare una maquina con los requisitos mínimos, es decir, 2 CPUs y 8GB de RAM. El espacio usado en disco para la instalación serán unos 25-30GB.

Por último, también necesitare un registro DNS que apunte a la maquina. Se podría instalar sin necesidad de un nombre dns usando un nombre falso para testeo.

Sistema operativo

- Rancher es compatible con cualquier sistema operativo linux que tenga instalado un clúster de kubernetes, que a su vez, se puede instalar prácticamente en cualquier distribución linux que sea compatible con Docker. En este proyecto voy a usar una maquina de Ubuntu 22.04 en su versión servidor, aunque como ya tenia preparada la instalación en Rocky Linux y la instalación es casi igual, indico los comandos correspondientes para cada sistema. Usare una instalación mínima ya que para instalar todo lo necesario solo me hace falta el sistema base y acceso con ssh.

Una vez instalado el sistema operativo procedo a empezar a instalar todos los paquetes necesarios para Rancher.

Configuración básica del firewall

- Tras acceder al servidor como root, lo primero que hay que hacer es securizar con un firewall el acceso a este servidor ya que va a estar expuesto a internet, para ello instalo e inicio el paquete de firewalld:

```
dnf install firewalld -y  
systemctl start firewalld
```

(En Ubuntu no hace falta instalar el firewall ya que usare ufw que ya viene instalado por defecto en el sistema)

Normalmente, antes de iniciar cualquier firewall habría que permitir primero el acceso ssh a el servidor para no quedarnos sin conexión con el servidor, pero la configuración por defecto de firewalld ya permite conexiones ssh. Para ufw si que hará falta permitir el servicio ssh antes de iniciar el firewall:

```
ufw allow from <IpPermitida> proto tcp to any port 22  
ufw enable  
systemctl status ufw
```

En este caso solo permitiré las ips de la empresa para que solo se pueda acceder al ssh desde dentro de la misma. En rocky linux también haré lo mismo pero creando una zona con las ips permitidas.

Para comprobar que se ha iniciado correctamente el firewall:

```
systemctl status firewalld
```

Por último, añado los servicios y puertos que me van a hacer falta para acceder a Rancher y limito el acceso ssh solo para la red interna.

```
firewall-cmd --permanent --add-service=http  
firewall-cmd --permanent --add-service=https  
firewall-cmd --remove-service=ssh --permanent  
firewall-cmd --new-zone=red-interna --permanent  
firewall-cmd --zone=red-interna --add-source=10.200.128.0/24 --permanent  
firewall-cmd --zone=red-interna --add-service=ssh --permanent  
firewall-cmd --zone=red-interna --add-port=6443/tcp --permanent  
firewall-cmd --reload
```

(Ubuntu)

```
ufw allow http  
ufw allow https  
ufw allow 6443  
ufw allow from 172.17.0.0/16  
ufw allow from 10.200.128.0/24
```

Como solo voy a usar un nodo no hace falta abrir muchos puertos.

Configuración clave ssh

- Para securizar todavía más el servidor, voy a registrar mi clave ssh publica para el usuario root y voy a deshabilitar la autentificación por contraseña. Para ello, creo el directorio .ssh en el directorio home de root y dentro creo un archivo llamado authorized_keys donde copiare la clave.

```
mkdir -p ~/.ssh && touch ~/.ssh/authorized_keys && chmod -R go= ~/.ssh
```

Una vez hecho esto y copiada la clave dentro del archivo, cambio la siguiente configuración del servicio ssh en "/etc/ssh/sshd_config" para desactivar el loggin con contraseña en texto plano.

```
# To disable tunneled clear text passwords, change to no here!  
PasswordAuthentication no  
#PermitEmptyPasswords no
```

Hecho esto ya solo podre acceder al servidor con una clave ssh.

Instalación

- A continuación voy a describir el proceso de instalación de Rancher desde un SO recién instalado.

Docker

- Lo primero que hay que instalar en el servidor sera docker, para ello tengo que añadir el repositorio correspondiente de docker para el SO. Al contrario que para Ubuntu, no hay un repositorio específico de docker para rocky linux, pero al estar basado en centos, es compatible con el repositorio de este.

```
dnf check-update  
dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

(Ubuntu)

```
apt update

apt install -y apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo "deb [arch=$(dpkg--print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

apt update
```

Una vez añadido el repositorio, antes de instalar docker, hay que tener en cuenta que la versión que estoy usando de rke no soporta versiones de docker superiores a la 20.10.x, así que tengo que instalar esa versión desde el repositorio, para ver las versiones disponibles en el repositorio que acabo de instalar uso el siguiente comando:

```
yum list docker-ce --showduplicates | sort -r
```

(Ubuntu)

```
apt list docker-ce -a
```

El número de versión será el de la segunda columna empezando este justo después de el doble punto ":" y terminando antes del guion "-", con lo cual la versión que tengo que instalar será la 20.10.24.

Ahora que ya tengo añadido el repositorio y sé qué versión tengo que instalar, hay que instalarlo e iniciar el servicio.

```
yum install -y docker-ce-20.10.24 docker-ce-cli-20.10.24 containerd.io
systemctl start docker
systemctl enable docker
```

(Ubuntu)

```
apt install -y docker-ce=5:20.10.24~3-0~ubuntu-jammy docker-ce-
cli=5:20.10.24~3-0~ubuntu-jammy containerd.io

systemctl start docker
systemctl enable docker
```

Compruebo que se ha iniciado correctamente

```
systemctl status docker
```

Por último, preparo un usuario que no sea root pero con permisos para ejecutar comandos de docker para utilizarlo como usuario de rancher y el clúster de kubernetes. Para ello creo el usuario y lo añado al grupo "docker".

```
adduser rancher
usermod -aG docker rancher
```

A este usuario deberé crearle un nuevo par de claves ssh y registrar su propia clave publica en su fichero "authorized_keys" para que más tarde el instalador tenga acceso.

Cluster de kubernetes (RKE)

- Ahora que docker esta instalado, tengo que instalar el clúster de kubernetes, he elegido instalar RKE (Rancher Kubernetes Engine) porque es una distribución de kubernetes creada por el equipo de rancher y es la recomendada para el mismo, aunque se puede instalar sobre cualquier otra distribución de kubernetes.

Lo primero sera descargar el fichero binario apropiado de rke, en este caso es el último release (1.4.4) en su versión de linux_amd ([Releases de RKE](https://github.com/rancher/rke/releases/download/v1.4.4/rke_linux-amd64)). A partir de ahora, una vez cambio a el usuario "rancher" ejecutare todos los comandos siguientes desde ese usuario.

```
dnf install -y wget
su - rancher
wget https://github.com/rancher/rke/releases/download/v1.4.4/rke\_linux-amd64
mkdir -p .local/bin && mv rke_linux-amd64 .local/bin/rke && chmod +x .local/bin/rke
```

(Ubuntu)

```
su - rancher
wget https://github.com/rancher/rke/releases/download/v1.4.5/rke\_linux-amd64
mkdir -p .local/bin && mv rke_linux-amd64 .local/bin/rke && chmod +x .local/bin/
echo 'export PATH="~/.local/bin:$PATH"' >> .bashrc
export PATH="~/.local/bin:$PATH"
rke
```

A continuación tengo que preparar la configuración del clúster dentro de un archivo llamado "cluster.yml". Voy a usar el [ejemplo de configuración minima](#) que proporciona rke y lo modiflico para que se adecué a mi servidor.

A parte de modificar la ip y el nombre del usuario, voy a añadir una configuración de ingress para activar el "ssl-pass-through", esto sera necesario para poder exponer públicamente a internet ciertas aplicaciones con ssl ya integrado una vez que este el clúster final funcionando.

```
nodes:
  - address: <ip_publica>
    user: rancher
    role:
      - controlplane
      - etcd
      - worker
    ssh_key_path: /home/rancher/.ssh/id_rsa
ingress:
  provider: nginx
  extra_args:
    enable-ssl-passthrough: true
```

Antes de instalar el clúster de rke debo crear una clave ssh para el usuario "rancher" de la misma forma que lo hice para root, para crearla simplemente uso el comando "ssh-keygen" con el usuario rancher y luego añado la clave publica generada en el fichero .ssh/authorized_keys

Ahora para crear el clúster de kubernetes ejecuto el siguiente comando mientras me encuentro en el mismo directorio en el que esta el fichero "cluster.yml"

```
rke up
```

Si todo ha ido correctamente se vera la siguiente linea al final del proceso:

```
INFO[0187] Finished building Kubernetes cluster successfully
```

Por último, en la documentación de RKE recomiendan hacer copias de los tres archivos que se han generado durante la instalación (cluster.rkestate, cluster.yml, kube_config_cluster.yml) ya que en estos ficheros tenemos toda la configuración y información de acceso del clúster. Ahora que ya tengo instalado y funcionando el clúster de RKE, hay que instalar las herramientas necesarias para administrarlo e instalar rancher las cuales son kubectl y helm. No hace falta que estas herramientas se instalen en el propio servidor de kubernetes ya que se puede administrar de manera remota, pero yo las voy a instalar directamente en el servidor.

Instalar Kubectl y Helm

- A partir de aquí hago todas las instalaciones con el usuario rancher.

Primero voy a instalar y configurar kubectl, para ello, tengo que elegir una versión y descargarla en el servidor con el comando "curl". Para elegir una versión hay que tener en cuenta que kubectl funciona con una versión inferior o superior de kubernetes que la suya, es decir, si instalamos kubectl 1.26, este nos valdrá para administrar clústeres con kubernetes 1.25.x, 1.26.x o 1.27.x

En el caso del clúster que he instalado tiene la versión 1.24, con lo cual debo instalar la versión 1.25 de kubectl. Ejecutando los siguientes comandos ya tendré descargado e instalado kubectl.

```
curl -LO https://dl.k8s.io/release/v1.25.0/bin/linux/amd64/kubectl
chmod +x kubectl
mkdir -p ~/.local/bin
mv ./kubectl ~/.local/bin/kubectl
```

Ahora tendré que copiar el archivo de conexión del clúster que se generó durante la instalación a el directorio de configuración de kubectl. Lo tendré que hacer con el usuario root ya que con rancher no tengo permisos para copiarlo.

```
mkdir ~/.kube
su -
cp /etc/rancher/rke2/rke2.yaml /home/rancher/.kube/config
chown rancher:rancher /home/rancher/.kube/config
su - rancher
```

Ya tengo configurado kubectl, ahora si ejecuto un "kubectl get nodes" debería de ver el nodo local.

NAME	STATUS	ROLES	AGE	VERSION
pmoldenhauer	Ready	control-plane, etcd, master	63m	v1.24.12+rke2r1

Ahora voy a instalar Helm, para ello, lo único que tengo que hacer es descargarme el fichero binario de la última versión de Helm de la [pagina oficial](#), descomprimirla, moverla a algún directorio que este añadido en \$PATH, en este caso lo voy a poner en ".local/bin/helm" y cambiarle el nombre al fichero a "helm".

```
wget https://get.helm.sh/helm-v3.11.3-linux-amd64.tar.gz
tar -zxvf helm-v3.11.3-linux-amd64.tar.gz
mv linux-amd64/helm .local/bin/helm
```

Hay muchas otras maneras de instalar helm, pero esta me ha parecido la más sencilla, sobre todo para más tarde hacerlo de manera automatizada. Ahora que tengo kubectl y helm listos, solo me queda instalar cert-manager y rancher en el clúster de RKE.

Cert-manager

- Cert-manager es un gestor de certificados que es necesario para instalar rancher si quiero tener certificados autofirmados o certificados de letsencrypt de manera automática. Para instalarlo, ya que es un chart de helm, lo único que hay que hacer es añadir el repositorio e instalarlo con helm.

Añado el repositorio y actualizo la información de los repos con los siguientes comandos:

```
helm repo add jetstack https://charts.jetstack.io
helm repo update
```

Por último instalo Cert-Manager:

```
helm install \
cert-manager jetstack/cert-manager \
--namespace cert-manager \
--create-namespace \
--version v1.11.0 \
--set installCRDs=true
```

Compruebo que se han iniciado correctamente los pods de cert-manager

```
kubectl get pods -n cert-manager
```

Y también uso el siguiente comando para comprobar que esta funcionando correctamente:

```
cmctl check api
```

Ya tengo todo listo para instalar rancher y poder acceder a la interfaz web.

Rancher

- Ya solo queda instalar el propio Rancher, para ello lo primero que hay que hacer es añadir el repositorio de Rancher que se quiera, hay tres opciones, el repositorio “stable”, que instalara una versión estable y muy testeada de rancher, el repositorio “latest”, que instalara la última versión de rancher disponible y el repositorio “alpha”, que instalara la última versión con todos los cambios ya disponibles que habrá en la siguiente versión. Yo he elegido instalar la versión estable, ya que esta suele ser la opción en un entorno de producción.

```
helm repo add rancher-stable https://releases.rancher.com/server-charts/stable  
helm repo update
```

Creo el namespace para rancher, debe tener el nombre "cattle-system"

```
kubectl create namespace cattle-system
```

Y por último, instalo rancher con helm, hay varias opciones para manejar los certificados de rancher, con certificados autofirmados, con letsencrypt o dándole ficheros de certificados previamente hechos. Para las dos primeras opciones necesitaremos cert-manager, por eso lo instale previamente ya que yo voy a usar la opción de letsencrypt con cert-manager.

Las opciones de instalación que pongo más abajo son:

1. El namespace donde se instalará rancher.
2. El hostname de rancher, el cual debe ser el nombre dns que apunta a el servidor.
3. La contraseña que se usará para acceder a la interfaz de rancher como administrador.
4. El tipo de manejo de certificados que usaremos, en este caso es letsencrypt.
5. El controlador de ingress que estemos usando, por defecto es nginx.

```
helm install rancher rancher-stable/rancher \  
--namespace cattle-system \  
--set hostname=pmoldenhauer.trevenque.es \  
--set bootstrapPassword=<pass> \  
--set ingress.tls.source=letsEncrypt \  
--set letsEncrypt.email=pmoldenhauer@trevenque.es \  
--set letsEncrypt.ingress.class=nginx \  
--set global.cattle.psp.enabled=false
```

Cuando acabe de instalar, uso el siguiente comando para ver el progreso del despliegue de rancher:

```
kubectl -n cattle-system rollout status deploy/rancher
```

```
Waiting for deployment "rancher" rollout to finish: 0 of 3 updated  
replicas are available...  
deployment "rancher" successfully rolled out
```

Y por último también se puede usar el siguiente comando para comprobar que todos los pods están iniciados y funcionando correctamente:

```
kubectl -n cattle-system get deploy rancher
```

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
rancher	3	3	3	3	3m

Una vez que haya acabado, ya estará rancher instalado y podre acceder a la interfaz web usando el FQDN que le haya asignado a rancher.

• Instalación de ArgoCD

- Para instalar ArgoCD, antes he tenido que conectar el clúster de harvester con rancher y he creado un clúster de kubernetes virtual, para enseñar como he hecho esto he puesto más adelante unas guiás de como lo he hecho en la sección de “Manuales de administración”.

Una vez conectado el clúster de harvester con rancher y creado el clúster con nodos virtuales de kubernetes ya puedo instalar argocd en el nuevo clúster creado.

Para poder instalar y iniciar sesión correctamente en argocd en el nuevo clúster que acabo de crear necesito tener instalado y configurado Kubectl para poder gestionar este clúster. Lo he instalado en el servidor de Rancher ya que este servidor tiene conexión con la red de VM donde están las maquinas virtuales de el nuevo clúster de kubernetes.

Para instalar kubectl lo hago igual que lo hice en mi maquina local durante la instalación de Rancher, el fichero "config" que debo usar es el que puedo descargar/copiar desde la pagina del clúster en rancher.

Una vez tengo Kubectl funcionando, para instalar la última versión de ArgoCD tengo que hacer lo siguiente:

1. Creo el namespace para argocd y despliego el proyecto en su última versión estable desde la pagina oficial

```
kubectl create namespace argocd
kubectl apply -n argocd -f
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

2. Instalo la CLI de argocd, esta me va a hacer falta para iniciar sesión por primera vez y también me servirá para gestionar argocd desde la terminal.

```
curl -sSL -o argocd-linux-amd64
https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64
sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd
rm argocd-linux-amd64
```

3. Expongo el servicio de ArgoCD como nodeport para poder acceder a la interfaz web. Más tarde, volveré a cambiar este servicio a "ClusterIP" y accederé a la interfaz de argoCD a través del reenvío con haproxy-ingress.

```
kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "NodePort"}}'
```

4. Compruebo la contraseña por defecto que se ha generado automáticamente para el primer inicio de sesión del usuario "admin", inicio sesión a través de la terminal y cambio la contraseña. Antes de iniciar sesión, tengo que comprobar en qué puerto se está ejecutando el servicio de ArgoCD.

- Voy al dashboard de rancher del clúster y busco el namespace de argocd.

The screenshot shows the ArgoCD dashboard for the 'argo-cluster' cluster. The top navigation bar includes 'Only User Namespaces', a search bar, and various icons. The main area displays the 'Namespace: argocd (Active)' details, showing it was created 21 hours ago and has the label 'kubernetes.io/metadata.name: argocd'. Below this, a summary bar shows 149 Active, 0 Transitioning, 0 Warning, 0 Error, and 0 Unknown resources. A table below lists resource types and their counts, with 'services' highlighted by a red arrow.

Type	Error	Warning	Transitioning	Active	Unknown	Total
apps.controllerrevisions	—	—	—	1	—	1
apps.deployments	—	—	—	6	—	6
apps.replicasets	—	—	—	6	—	6
apps.statefulsets	—	—	—	1	—	1
argoproj.io.appprojects	—	—	—	2	—	2
configmaps	—	—	—	8	—	8

- Bajo en la lista de "Resource" y pulso en "Services".

The screenshot shows the 'Resources' tab of the ArgoCD dashboard. The summary bar at the top shows 149 Active, 0 Transitioning, 0 Warning, 0 Error, and 0 Unknown resources. The main table lists various Kubernetes resources, with the 'services' row highlighted by a red arrow. The 'services' row shows 8 Active resources.

Type	Error	Warning	Transitioning	Active	Unknown	Total
apps.controllerrevisions	—	—	—	1	—	1
apps.deployments	—	—	—	6	—	6
apps.replicasets	—	—	—	6	—	6
apps.statefulsets	—	—	—	1	—	1
argoproj.io.appprojects	—	—	—	2	—	2
configmaps	—	—	—	8	—	8
discovery.k8s.io.endpointslices	—	—	—	8	—	8
endpoints	—	—	—	8	—	8
events	—	—	—	31	—	31
events.k8s.io.events	—	—	—	31	—	31
networking.k8s.io.networkpolicies	—	—	—	7	—	7
pods	—	—	—	7	—	7
rbac.authorization.k8s.io.roles	—	—	—	5	—	5
rbac.authorization.k8s.io.rolebindings	—	—	—	8	—	8
secrets	—	—	—	4	—	4
services	—	—	—	8	—	8
serviceaccounts	—	—	—	8	—	8

- Aquí busco el servicio que se llama "argocd-server", a la derecha puedo ver los dos puertos en los que se está ejecutando argocd en http y https, como ip puedo usar cualquiera de las ips de los nodos del clúster ya que al haber puesto el servicio en modo "NodePort" el ingress del clúster se encarga de reenviar las conexiones de un servidor a otro. Si pulso en el enlace que tiene cualquiera de los dos puertos me lleva a la página de ArgoCD, siempre y cuando mi equipo tenga acceso a la red que en la que están los nodos del clúster.

State	Name	Target	Selector	Type	Age
(Active)	argocd-applicationset-controller	10.43.1.209:7000 → 5556/TCP metrics/metrics/TCP	app.kubernetes.io/name=argocd-applicationset-controller	Cluster IP	21 hours
(Active)	argocd-dex-server	10.43.201.133:5557 → 5557/TCP 10.43.201.133:5558 → 5558/TCP	app.kubernetes.io/name=argocd-dex-server	Cluster IP	21 hours
(Active)	argocd-metrics	10.43.233.100:8082 → 8082/TCP	app.kubernetes.io/name=argocd-application-controller	Cluster IP	21 hours
(Active)	argocd-notifications-controller-metrics	10.43.222.93:9001 → 9001/TCP	app.kubernetes.io/name=argocd-notifications-controller	Cluster IP	21 hours
(Active)	argocd-redis	10.43.204.76:6379 → 6379/TCP	app.kubernetes.io/name=argocd-redis	Cluster IP	21 hours
(Active)	argocd-repo-server	10.43.56.69:8081 → 8081/TCP 10.43.56.69:8084 → 8084/TCP	app.kubernetes.io/name=argocd-repo-server	Cluster IP	21 hours
(Active)	argocd-server	31740/TCP 32759/TCP	app.kubernetes.io/name=argocd-server	Node Port	21 hours
(Active)	argocd-server-metrics	10.43.158.134:8083 → 8083/TCP	app.kubernetes.io/name=argocd-server	Cluster IP	21 hours

- Ahora consigo el token de inicio de sesión/contraseña inicial e inicio sesión.

```
argocd admin initial-password -n argocd
```

```
gtk@pmoldenhauer:~$ argocd admin initial-password -n argocd
mYeScpbuiK3sxajK
```

```
This password must be only used for first time login. We strongly recommend you update the password using `argocd account update-password`.
```

```
argocd login 10.200.129.170:32759
```

```
gtk@pmoldenhauer:~$ argocd login 10.200.129.170:32759
WARNING: server certificate had error: x509: cannot validate certificate for 10.200.129.170 because it doesn't contain any IP SANs. Proceed insecurely (y/n)? y
Username: admin
Password:
'admin:login' logged in successfully
Context '10.200.129.170:32759' updated
```

- Por último cambio la contraseña.

```
argocd account update-password
```

5. Ya tengo instalado ArgoCD y puedo acceder a su interfaz web con el usuario admin y la contraseña que acabo de cambiar, usando cualquiera de las ips de los workers del nodo con el puerto que vi antes en rancher.

• Instalación de Haproxy

- Para poder acceder a las aplicaciones que desplegare en el clúster de argocd desde internet, no puedo hacerlo directamente desde el clúster, ya que no tengo un direccionamiento publico en ese clúster o en el clúster de harvester. Por eso, para poder sortear este problema, voy a acceder a las aplicaciones de argocd redireccionando el trafico a través de la maquina virtual de rancher que si tiene direccionamiento publico. Normalmente, para hacer esto se crearía una maquina virtual a parte para hacer esto exclusivamente, pero para este proyecto voy a aprovechar la maquina virtual y el clúster de RKE que ya tengo y voy a desplegar en este un deployment y service de haproxy.

Para esto voy a hacer uso de un nuevo dominio DNS (pmolden.es) en el cual puedo crear subdominios y de un software de balanceo de carga, haproxy.

Archivo de configuración de Haproxy

- El archivo de configuración de haproxy tiene una sintaxis muy estricta, hay que respetar la indentación y siempre acabar el archivo con un salto de linea vacío.

```
defaults
  mode tcp
  timeout client 10s
  timeout connect 5s
  timeout server 10s

frontend http
  bind *:80
  default_backend httpservers

frontend https
  bind *:443
#  option tcplog
  default_backend httpservers

backend httpservers
  balance roundrobin
  # añado solo los nodos workers
  server server1 10.200.129.166:80
  server server2 10.200.129.168:80
  server server3 10.200.129.169:80

backend httpservers
  mode tcp
  balance roundrobin
  option ssl-hello-chk
  # añado solo los nodos workers
  server server1 10.200.129.166:443 check
  server server2 10.200.129.168:443 check
  server server3 10.200.129.169:443 check
```

Con esta configuración le estoy diciendo a haproxy que pase todo el trafico tcp que entre por el puerto 80 o 443 de cualquier dirección ip del pod de haproxy, que en este caso sera una ip privada conectada a través de un servicio ClusterIP, a las ips de las maquinas virtuales de los nodos worker del clúster de argocd en sus respectivos puertos http y https.

Despliegue de Haproxy

- Haproxy es un software de balanceo de carga opensource muy utilizado tanto en entornos empresariales como a nivel usuario, a parte de la opción de opensource, los desarrolladores de haproxy tienen opciones de pago con soporte y con más prestaciones para empresas. En el caso de este proyecto voy crear un deployment de kubernetes usando el contenedor de docker oficial de haproxy.

Mi intención es reenviar todo el trafico que llegue a el servidor de rancher pero dirigido a los distintos subdominios de "pmolden.es" que haya configurado en el ingress del clúster a las maquinas virtuales del clúster de argocd. Para ello, lo que debo hacer es crear un deployment y un service para haproxy, esto lo he hecho a través de la interfaz de Rancher en el clúster local, he creado un project y namespace nuevos en el clúster para tener separado el haproxy del sistema de rancher.

Project Namespace	Project	Status	Last Update
default	Project: Default	Active	17 days
haproxy	Project: haproxy	Active	1 day
System	Project: System		

Una vez creados el project y namespace para crear el deployment voy a la pestaña de "Deployments" del menú de "Workloads" de rancher y pulso el botón de crear arriba a la derecha.

Para crear el deployment hay que configurar varias cosas, lo primero seria darle un nombre y descripción al deployment y decirle cuantas replicas va a tener. Le cambio el nombre de pod, la imagen de contenedor que voy a usar, que es la última disponible de haproxy, y selecciono la opción para que solo se descargue la imagen del contenedor si no esta ya disponible.

The screenshot shows the Argo CD interface for creating a new Deployment. The left sidebar shows the cluster structure with 'Deployments' selected. The main panel is titled 'Deployment: haproxy [Active]'. The 'General' tab is active, showing the namespace as 'haproxy', name as 'haproxy', and a description: 'Servicio de balanceo de cargas con haproxy para reenviar trafico a el cluster de argocd.'. The 'Replicas' field is set to 2. The 'Image' section specifies 'Container Image: haproxy' and 'Pull Policy: IfNotPresent'. The 'Networking' section is collapsed. At the bottom right are buttons for 'Cancel', 'Edit as YAML', and 'Save'.

Lo siguiente que hago es limitar los recursos de los pods, en este caso como no voy a tener mucho trafico de red le pondré un limite bajo de cpu y memoria ram, aunque incluso con estos limites, al ser un contenedor y solo tener que redireccionar trafico, podría soportar bastantes conexiones.

The screenshot shows the Argo CD interface with the same deployment configuration, but now with resource limits applied. The 'Resources' section in the 'General' tab includes 'CPU Reservation: 250 mCPUs', 'Memory Reservation: 512 MiB', 'CPU Limit: 500 mCPUs', and 'Memory Limit: 1024 MiB'. The 'Networking' section remains collapsed. The bottom right buttons are 'Cancel', 'Edit as YAML', and 'Save'.

Rancher da la opción de crear un servicio que apunte a el deployment directamente desde este menú de creación, así que voy a crear un servicio para luego usarlo con ingress, el servicio que necesitare sera uno de tipo ClusterIP escuchando en dos puertos, el 80 para http y el 443 para https.

The screenshot shows the Rancher interface for creating a new service. On the left, there's a sidebar with various cluster metrics and resource counts. The main area is titled 'Networking' and contains fields for defining a service. Two services are listed: 'haproxy-http' and 'haproxy-https'. Each entry includes a dropdown for 'Service Type' (set to 'Cluster IP'), a 'Name' field, a 'Private Container Port' (80 for http, 443 for https), a 'Protocol' dropdown (set to 'TCP'), and buttons for 'Add Host' and 'Remove'.

Por último, tengo que compartir el fichero de configuración de haproxy de alguna manera para que este disponible en todos los pods y, en caso de tener que modificarlo, no tener que hacerlo a mano en cada uno de ellos. Para ello, he guardado la configuración como un configmap que luego he montado directamente como fichero en el directorio del contenedor donde debe ir esta configuración. Si tuviese que cambiar alguna ip o opción, solo tendría que cambiarla en el configmap y renovar los pods.

This screenshot shows the creation of a ConfigMap named 'haproxy-config' in the 'haproxy' namespace. The interface allows setting the namespace, name, and description. Under the 'Data' tab, a key 'haproxy.cfg' is defined with its corresponding configuration text. Buttons for 'Add' and 'Read from File' are visible at the bottom.

```

defaults
mode tcp
timeout client 10s
timeout connect 5s
timeout server 10s

frontend http
bind *:80
default_backend httpservers
  
```

Deployment: haproxy (Active)

Namespace: haproxy Age: 15 days Pod Restarts: 0

Namespace: haproxy Name: haproxy Description: Servicio de balanceo de cargas con haproxy para reenviar trafico a el cluster de argosc.

Replicas: 2

Deployment Pod haproxy-0 + Add Container

Storage

haproxy-conf (ConfigMap)

Mount Point: /usr/local/etc/haproxy Sub Path in Volume: Read Only Remove

Add Mount Select Volume

Cluster Tools v2.7.3 Cancel Edit as YAML Save

Deployment: haproxy (Active)

Namespace: haproxy Age: 15 days Pod Restarts: 0

Namespace: haproxy Name: haproxy Description: Servicio de balanceo de cargas con haproxy para reenviar trafico a el cluster de argosc.

Replicas: 2

Deployment Pod haproxy-0 + Add Container

Labels & Annotations

Networking

Node Scheduling

Pod Scheduling

Resources

Scaling and Upgrade Policy

Security Context

Storage

ConfigMap

Volume Name: haproxy-conf Default Mode: 644

ConfigMap: haproxy-config Optional: Yes No

Add Volume

Cluster Tools v2.7.3 Cancel Edit as YAML Save

Una vez hechas estas configuraciones, el resto de opciones las dejo por defecto.

Ahora ya tengo las dos replicas del pod de haproxy funcionando, solo me falta configurar ingress para que reenvíe el trafico a estos pods.

The screenshot shows the Rancher interface for managing a Kubernetes cluster. On the left, a sidebar lists various cluster resources: Cluster (selected), Workloads (CronJobs: 0, DaemonSets: 2, Deployments: 13, Jobs: 6, StatefulSets: 0, Pods: 24), Apps, Service Discovery, Storage, Policy, and More Resources. The main panel displays a deployment named "haproxy" (Active) in the "local" namespace. It shows 2 replicas running, both with the image "haproxy" and IP addresses 10.42.0.180 and 10.42.0.179 respectively. The deployment has an age of 1.1 days and 0 pod restarts. Endpoints are listed as 443/HTTPS, 80/HTTP. Annotations include "Show 1 annotation". Below the deployment details, a table shows the "Pods by State" section with 2 Running pods. The table columns are: State (Running), Name (haproxy-7b8dd9969-fjsdd, haproxy-7b8dd9969-vdp4q), Image (haproxy), Ready (1/1, 1/1), Restarts (0, 0), IP (10.42.0.180, 10.42.0.179), Node (217.18.163.165, 217.18.163.165), and Age (25 mins, 25 mins).

En el repositorio de github del proyecto he subido los dos archivos yaml del deployment y service que genera rancher.

Manuales de administración

- Al igual que con los manuales de instalación, voy a separar en los distintos procesos necesarios para cada parte del proyecto en varios manuales.

• Conectar el clúster de Harvester con Rancher

- Una vez que tanto Rancher como el clúster de Harvester están funcionando, puedo conectar el clúster a Rancher para poder gestionarlo desde ahí y crear de manera fácil y casi automatizada clusteres de kubernetes dentro de Harvester. Para hacer esto, lo primero que hay que hacer es importar un clúster en rancher en la pestaña del menú de la izquierda "Virtualization Manager".

The screenshot shows the Rancher interface with the "Virtualization Manager" tab selected in the sidebar. The main area displays a "Welcome to Rancher" message with a green landscape illustration. Below it, a table shows the imported cluster "local" (RKE2). The table columns are: Provider (Local RKE2), Kubernetes Version (v1.24.12+rke2r1), CPU (4 cores), Memory (7.76 GiB), and Pods (26/110). There are buttons for Manage, Import Existing, Create, and Filter. To the right, there is a "Links" sidebar with links to Docs, Forums, Slack, File an issue, Get Started, and Commercial Support. At the bottom, there are "Get Support" and "About" links.

Dentro de esta pestaña, hay que elegir la opción de "Import" que está arriba a la derecha.

The screenshot shows the 'Harvester Clusters' page under the 'Virtualization Management' section. A red box highlights the 'Import Existing' button in the top right corner. Below it, there's a green icon with a gear and a network symbol. Text at the bottom explains Harvester is a modern Hyperconverged Infrastructure (HCI) solution built for bare metal servers using enterprise-grade open source technologies including Kubernetes, Kubevirt and Longhorn. A link to the Harvester Web Site and Docs is provided.

Se abre una pagina donde hay que poner un nombre para el clúster y puedo elegir que miembros tienen acceso a este clúster, los miembros son los distintos usuarios que se pueden crear en rancher para separar los permisos a los distintos clusteres y opciones de Rancher. Una vez puesto el nombre elijo la opción de "Create" de abajo a la derecha.

The screenshot shows the 'Harvester Cluster: Create' form. It includes fields for 'Cluster Name' (set to 'harvester-test-cluster') and 'Cluster Description'. On the left, there are tabs for 'Member Roles', 'Agent Environment Vars', and 'Labels & Annotations'. Under 'Member Roles', there's a table with a row for 'User' (with 'Default Admin (admin)' and 'Local' options), a 'Role' column (set to 'Cluster Owner'), and an 'Add' button. At the bottom right of the form are 'Cancel', 'Edit as YAML', and a red-highlighted 'Create' button.

Por último, se abre la pestaña del clúster donde tengo las instrucciones para unir el clúster de harvester. Para esto, hay que copiar el enlace que aparece aquí y ir a la interfaz web del clúster de Harvester.

The screenshot shows the Harvester Cluster interface for a cluster named "harvester-test-cluster". The status is "Pending". A message indicates the cluster is transitioning. The "Registration" tab is selected, showing steps to register the cluster:

- Go to the Advanced / Settings page of the target Harvester's UI.
- Find the cluster-registration-url setting and click the : > Edit Setting button.
- Input the following registration URL and click the Save button.

The registration URL is listed as https://pmoldenhauer.trevende.es/v3/import/4fbvj21j9bckz67q1ffcpr4s5q4x4s9m7bzz7kdh7qfxpc6kdpjgg_c-m-q87r9rt9.yaml.

v2.7.2

En la interfaz de Harvester hay que ir a "Advanced->Settings".

The screenshot shows the Harvester Cluster dashboard for "local". The cluster version is v1.1.1 and it was created 5 days ago. The dashboard displays the following information:

- Hosts:** 1 Host, 2 Images
- Virtual Machines:** 1 Virtual Machine, 2 Volumes
- VM Networks:** 1 VM Network
- Capacity:**
 - CPU:** Reserved: 6.32 / 8, Used: 2.56 / 8, Utilization: 31.98%
 - Memory:** Reserved: 9.74 / 142 GB, Used: 12 / 142 GB, Utilization: 8.45%
 - Storage:** Reserved: 264 / 879 GB, Used: 2.94 / 879 GB, Utilization: 0.33%
- Metrics:** Cluster Metrics (selected), VM Metrics. Includes CPU Utilization, Load Average, and Memory Utilization graphs.

Support v1.1.1 English

The screenshot shows the Harvester Settings page. On the left sidebar, the 'Settings' option is selected and highlighted with a red arrow. The main area displays several configuration options:

- additional-ca**: Custom CA root certificates for TLS validation. A 'Show additional-ca' button is present.
- auto-disk-provision-paths**: Specify the disks (using glob pattern) that Harvester will automatically add as VM storage. The value is currently '<None>'.
- backup-target**: Custom backup target to store VM backups. Buttons for 'Show backup-target' and 'Test connection' are available.
- cluster-registration-url** (Modified): Registration URL for multi-cluster management. The value is 'https://pmoldenhauer.trevenque.es/v3/import/h6b8z12fff9bgxdr4grvzsqpgd5n28gw587nhk5jhfw6qd4birm4l_c-m-6wg8n662.yaml'. An 'Edit Setting' button is located to the right.
- containerd-registry**: A 'Show containerd-registry' button is present.

Y aquí buscar la opción de "cluster-registration-url" y la editarla para poner la url que copiada de Rancher.

This screenshot shows a close-up of the 'cluster-registration-url' setting in Harvester. The URL 'https://pmoldenhauer.trevenque.es/v3/import/h6b8z12fff9bgxdr4grvzsqpgd5n28gw587nhk5jhfw6qd4birm4l_c-m-6wg8n662.yaml' is displayed, indicating it has been modified. An 'Edit Setting' button is visible at the top right.

Si todo ha salido correctamente, en pocos segundos podremos ver que el clúster ya esta añadido y activo en el menú "Virtualization Manager" de Rancher y podre acceder a la interfaz web de Harvester pulsando sobre el nombre del clúster.

The screenshot shows the Rancher Virtualization Management interface. In the left sidebar, the 'Harvester Clusters' tab is selected. The main area displays a table of clusters:

State	Name	Version	Machines	Age	Actions
Active	molden-harvester	v1.24.7+rke2r1	3	29 mins	<button>Manage</button>

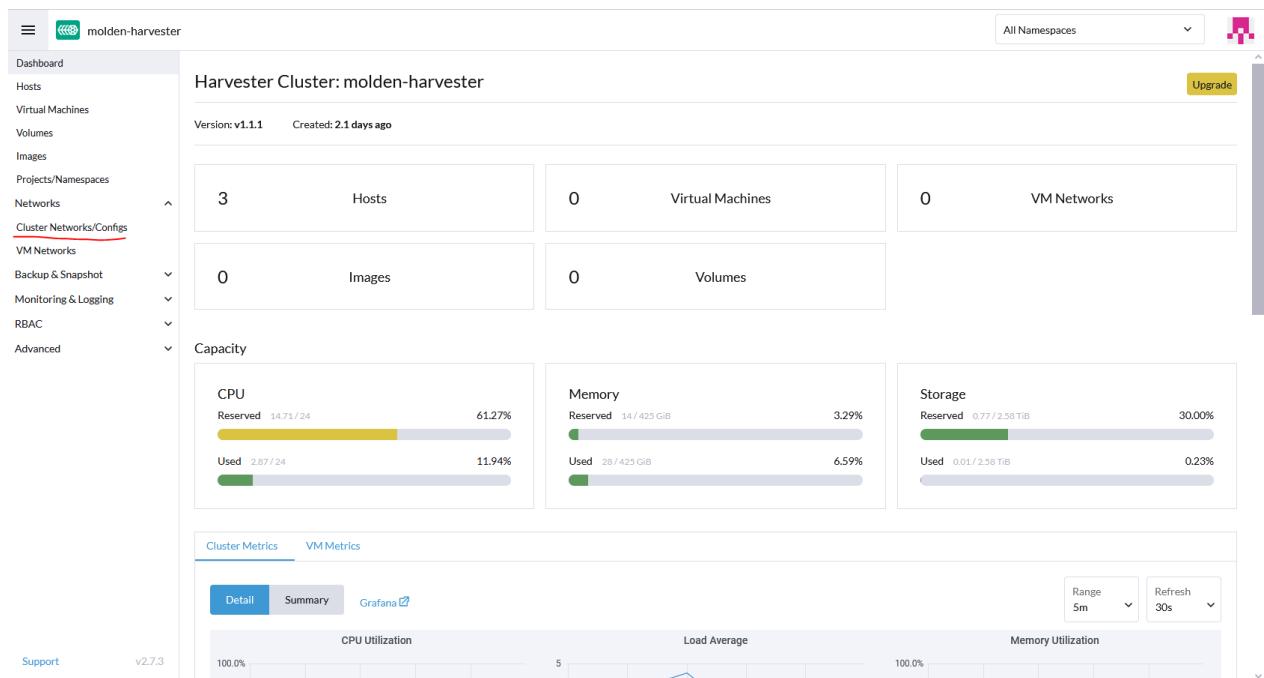
- **Configuración de las redes en Harvester**

Creación de las redes

- Una vez está el clúster funcionando con los tres servidores añadidos hay que configurar el rango de interfaces que se van a utilizar para cada red. En Harvester podemos configurar tantas redes distintas como queramos haciendo uso de vlans. La red de Management se crea automáticamente al crear el clúster.

Lo primero que voy a hacer es crear las redes y asignarles las interfaces físicas que va a usar cada una, el proceso para la creación de la red de VM y la de Storage es el mismo.

Para crear las redes hay que abrir la interfaz web de Harvester, ya sea directamente con la dirección del clúster o a través de Rancher, desde aquí tengo que ir a la opción del menú "Cluster Networks/Configs".



Aquí se pueden ver todas las redes que tengo creadas en el clúster, ahora mismo solo tengo la red de Management. Creo una nueva con el botón de arriba a la derecha "Create cluster network", al pulsar, se abre una pestaña donde podre ponerle un nombre y una pequeña descripción a la red, también se pueden añadir etiquetas o anotaciones. Para terminar de crear la red pulso el botón de abajo a la derecha "Create".

Cluster Network: Create

Name*: Storage

Description: Red usada para el intercambio de datos del almacenamiento de Longhorn

Labels

Add Label

Annotations

Add Annotation

Cancel Edit as YAML Create

Ya esta creada la red, pero ahora hay que crear una configuración para la misma, para ello, en la ventana anterior hay un botón al lado de la red "Create Network Config".

Cluster Networks/Configs

Create Cluster Network

Filter

State	Name	Type	Age
	Cluster Network: mgmt		
mgmt is a built-in cluster management network and does not support any additional network configurations.			
	Cluster Network: storage		
There are no network configs defined.			

Create Network Config

Aquí podremos darle un nombre y descripción a la configuración y modificar varias opciones de la red, en este caso las que hay que modificar están en la pestaña de la izquierda "Uplink", aquí se asignan las interfaces de los servidores que se van a usar en esta red y se define el modo de bonding que se va a usar, como en el switch esta configurado en modo LACP aquí hay que elegir la opción 802.3ad, que es el el termino que se usa para la agregación de enlaces con negociación, LACP. Si las interfaces de los distintos nodos tienen los mismos nombres y están disponibles en todos no tendremos problemas al crear la configuración, de lo contrario pedirá que configuremos manualmente las interfaces de cada nodo.

The screenshot shows the 'molden-harvester' interface with the 'Network Config: Create' screen open. The left sidebar shows various project and namespace options. The main form has a 'Name*' field set to 'storage-conf' and a 'Description' field with placeholder text. The 'Node Selector' section is expanded, showing an 'Uplink' tab selected. Under 'NICs*', four network interfaces are listed: 'ens3f0 (Up)', 'ens3f1 (Up)', 'ens3f2 (Up)', and 'ens3f3 (Up)'. Below these, there's a 'Bond Options' section with a 'Mode' dropdown set to '802.3ad' (highlighted by a red arrow). Other bond modes like 'RoundRobin' and 'LACP' are also shown. At the bottom right of the form are 'Cancel', 'Edit as YAML', and 'Create' buttons.

Ya esta creada la red y esta disponible para usarla como red para maquinas virtuales o storage, para crear la otra red el proceso es exactamente el mismo.

Asignar la red de storage

Una vez creada la red de storage, hay que cambiar la configuración del clúster de harvester para que la use como tal. Para ello, hay que ir a la pestaña del menú de la izquierda "Advanced -> Settings", aquí hay una lista de las distintas configuraciones del clúster que se pueden cambiar, entre ellas la de "storage-network". Para modificarla pulso en los puntitos de la derecha de la opción y elijo "modify".

Una vez dentro de la configuración de la red de storage hay que cambiar varias cosas, lo primero es pulsar en "Enabled" para que harvester use una red de storage, por defecto usa la red de management si no esta habilitada esta opción, lo siguiente es elegir una id de Vlan que vamos a usar, en este caso sera la 130, después elijo la red del clúster que se va a usar, la cual sera la red de storage que he creado anteriormente y por último tengo que asignarle un rango de Ips que va a usar para esta red, como va a ser solo una red interna que no se va a comunicar hacia fuera usare un rango de ips privado, 192.168.1.0/24. Pulsamos en el botón de "Save" de abajo a la derecha y ya tenemos configurada y funcionando la red de storage.

Asignar la red para las maquinas virtuales

Lo último que hay que hacer para terminar de configurar las redes de Harvester sera asignar la red de Vm que acabo de crear para que este disponible como tal a la hora de crear maquinas virtuales o clusteres de kubernetes. Para ello, en el menú de la izquierda hay que ir a "Networks -> VM Networks".

Aquí elijo la opción de "Create" de arriba a la derecha y se habré una ventana donde se pueden elegir varias opciones. En este caso hay que modificar varias, lo primero sera ponerle un nombre y descripción a la red virtual, lo siguiente es elegir el tipo de red, sera una red vlan taguada, elijo una id para la vlan, 129, y por último elijo la red del clúster que se va a usar, vm-net. En la pestaña "Route" de la izquierda podemos cambiar las opciones de ruteo de la red, en este caso lo dejare automático por dhcp ya que hay un dhcp funcionando en la red, pero podría poner el gateway y el rango de ips de manera manual. Pulso el botón de "Create" De abajo a la derecha.

En el menú anterior veremos la nueva red vm creada, si esperamos unos segundos debería de ver el estado "Active" en la columna de "Route connectivity", esto significa que la red esta funcionando correctamente.

Name	Type	Cluster Network	Vlan ID	Route Connectivity	Age
vm-network	L2VlanNetwork	vm-net	129	Active	34 secs

- Creación del cluster de Kubernetes en Rancher sobre el cluster de Harvester**

- Una vez que ya tengo tanto el clúster de Harvester creado y configurado como el servidor de Rancher funcionando y conectado con Harvester, puedo crear clusteres de kubernetes en nodos virtuales desde rancher alojados en el clúster de Harvester de manera muy sencilla.

Preparación de la imagen para los nodos

- Antes de crear el clúster, debo de subir una imagen con una maquina virtual con un sistema operativo linux a el clúster de harvester para poder elegirla como base para crear los nodos del nuevo clúster de kubernetes. Para esto puedo crearla yo mismo o usar una de las imágenes de maquina virtual que nos proporcionan muchos de los SO linux en sus paginas oficiales.

Para el ejemplo subir una imagen a Harvester, voy a usar la imagen "CloudGeneric-Base" de Rocky Linux 9.1, la cual la proporciona el propio equipo de rocky linux en su pagina oficial, el proceso para subir cualquier otra imagen de SO sera exactamente el mismo.

Lo primero para crear la imagen es ir al menú de "Images" de la interfaz web de Harvester y ahí pulsar el botón de "Create" de arriba a la derecha.

State	Name	Progress	Size	Age
There are no rows to show.				

Aquí hay que poner un nombre y descripción para la imagen y también un namespace, he creado uno nuevo para alojar todas las maquinas e imagenes del clúster de kubernetes. Hay dos opciones a la hora de subir la imagen, usar una URL desde la cual harvester descargara la imagen o subir una imagen desde el equipo local, yo voy a poner directamente la URL de la pagina oficial de la versión de la imagen que necesito. Para terminar, pulso el botón de "Create" de abajo a la derecha.

Una vez creada la imagen se empezara a descargar y la guardara en el almacenamiento de Harvester.

Con esto ya tendría la imagen disponible para hacer el clúster de kubernetes desde rancher, pero antes voy a probarla haciendo una maquina virtual en harvester para comprobar que todo funciona correctamente.

Creación de maquina virtual en Harvester (Prueba de la imagen)

- Para crear una maquina virtual en Harvester hay que ir al menú de "Virtual Machines" del

dashboard de Harvester y ahí pulsar el botón de "Create" de arriba a la derecha.

The screenshot shows the Harvester dashboard interface. On the left, there's a sidebar with various navigation options: Dashboard, Hosts, Virtual Machines (which is currently selected and highlighted in red), Volumes, Images, Projects/Namespaces, Networks, Backup & Snapshot, Monitoring & Logging, RBAC, and Advanced. At the bottom of the sidebar, it says 'Support v2.7.3'. The main content area is titled 'Virtual Machines'. It features a table with the following columns: State, Name, CPU, Memory, IP Address, Node, and Age. A message at the bottom of the table states 'There are no rows to show.' In the top right corner of the main area, there's a blue 'Create' button. Above the main content area, there's a dropdown menu labeled 'All Namespaces' and a small icon.

Aquí tendremos que especificar tanto un nombre, descripción y namespace para la maquina como las características de la misma, no es necesario elegir el mismo namespace que el de la imagen para poder usarla. Harvester tiene varias plantillas pre-hechas para las maquinas virtuales, yo voy a usar una especifica para imágenes "raw" que vale también para "qcow2" y la voy a modificar un poco.

También es importante añadir una clave ssh, ya que generalmente las imágenes cloud están configuradas para poder solo iniciar sesión con clave publica. Al añadir una clave, harvester la añadirá a la maquina durante la creación de la misma para el usuario root o para el usuario que haya sido asignado en el sistema operativo como usuario por defecto, normalmente este usuario sera un usuario con el nombre del sistema operativo (debian, ubuntu, etc.) o el usuario cloud-user, en cualquier caso, siempre estará indicado en la pagina desde donde se haya descargado la imagen.

Virtual Machine: Create

Namespace* **kubecuster**

Name* **rocky-test**

Description
Any text you want that better describes this resource

Use VM Template:

Template **harvester-public/raw-image-base-template**

Version **1 (Default)**

Basics

CPU* **2**

Memory* **4** GiB

SSHKey **kubecuster/pedro-pc-empresa**

Cancel Edit as YAML Create

En la pestaña "Volumes" elijo la imagen que he creado antes de rocky linux dentro de el volumen root. En este menú podría crear nuevos volúmenes para la maquina virtual si quisiese.

Image Volume	
Name* rootdisk	Type disk
Image* kubecuster/rocky-linux	Size* 15 GiB
Bus VirtIO	
bootOrder: 1	
<input type="button" value="Add Volume"/> <input type="button" value="Add Existing Volume"/> <input type="button" value="Add VM Image"/> <input type="button" value="Add Container"/>	
<input type="button" value="Cancel"/> <input type="button" value="Edit as YAML"/> <input type="button" value="Create"/>	

Por último, es importante cambiar la red default por la red de vm que he preparado para las maquinas virtuales para que funcione en modo bridge y este conectada directamente a la red, aunque podría dejarla por defecto y usar la red de management y funcionaria sin problema en modo nat.

The screenshot shows the 'Networks' configuration page in Rancher. On the left, a sidebar lists 'Basics', 'Volumes', 'Networks' (which is selected), 'Node Scheduling', and 'Advanced Options'. The main area is titled 'Networks' and contains a form for creating a new network. The 'Name*' field is set to 'default', 'Model*' is 'virtio', 'Network*' is 'default/vm-network', and 'Type*' is 'bridge'. Below the form is a 'Mac Address' input field with a 'pencil' icon. At the bottom left is a blue 'Add Network' button, and at the bottom right are 'Cancel', 'Edit as YAML', and a large blue 'Create' button.

Después de uno o dos minutos en los que tarda en iniciar la maquina y recibir una ip por dhcp, puedo acceder a ella con mi clave ssh sin problema.

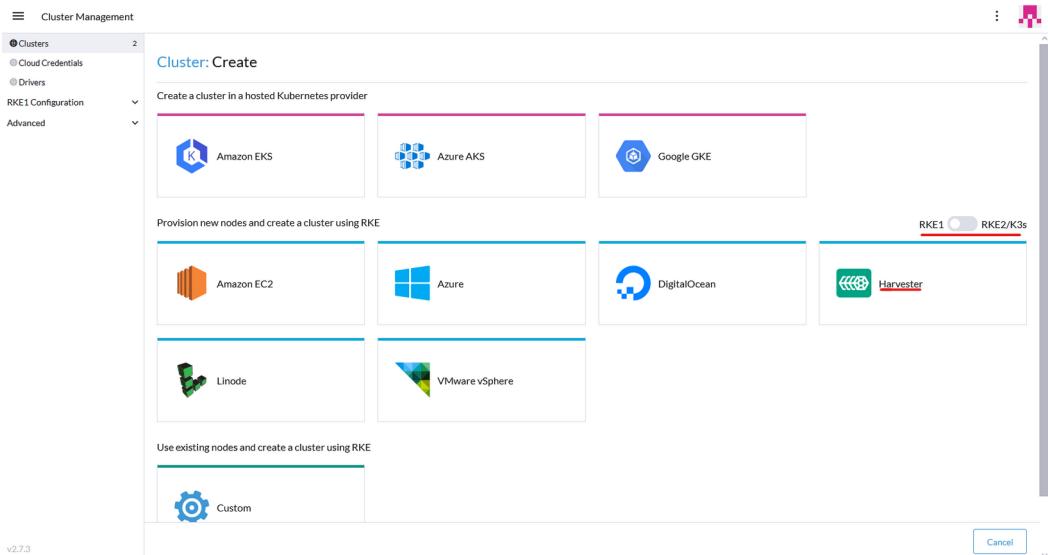
The screenshot shows the 'Virtual Machines' section of the Rancher dashboard. The left sidebar includes 'Dashboard', 'Hosts', 'Virtual Machines' (selected), 'Volumes', 'Images', 'Projects/Namespace', 'Networks', 'Backup & Snapshot', 'Monitoring & Logging', 'RBAC', and 'Advanced'. The main area shows a table of virtual machines. A single row is selected for 'rocky-test' in the 'kubecluster' namespace, which is 'Running' on node 'molden-harvester-02' with 2 CPU and 4 GiB memory. At the top right is a blue 'Create' button.

Creación del clúster

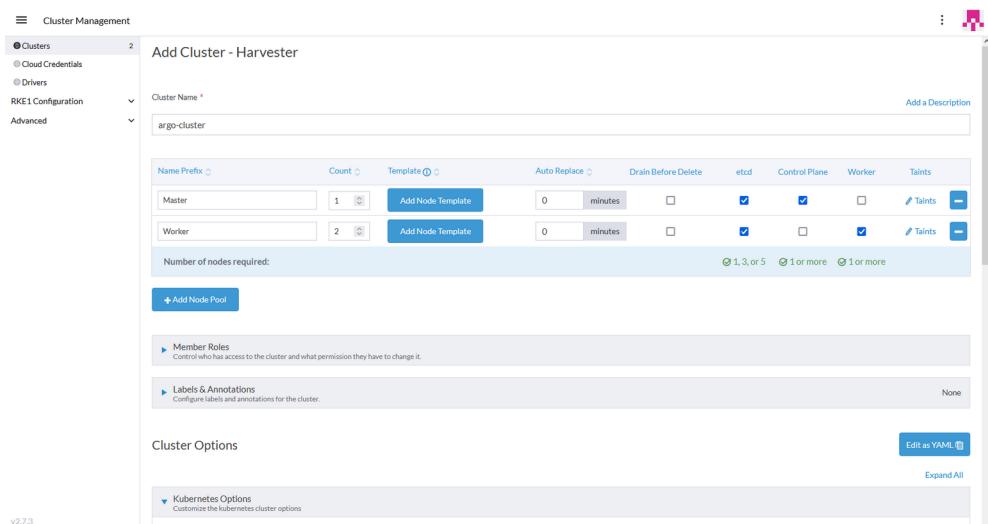
- Una vez creada la imagen del SO base, la cual finalmente sera una de debian 11, se puede crear el clúster de kubernetes. En el dashboard, donde aparece el clúster local donde esta instalado rancher, se puede crear un nuevo clúster pulsando en el botón "Create".

The screenshot shows the 'Clusters' section of the Rancher dashboard. The left sidebar includes 'Clusters' (with 1 entry), 'Manage', 'Import Existing', 'Create' (highlighted in blue), and 'Filter'. The main area shows a table of clusters. One cluster, 'local', is listed with 'State' as 'Active', 'Name' as 'local', 'Provider' as 'Imported', 'Kubernetes Version' as 'v1.25.9', 'CPU' as '4 cores', 'Memory' as '7.67 GiB', and 'Pods' as '22/110'. At the top right is a blue 'Create' button.

Al pulsar, se abre una ventana donde puedo elegir desde donde quiero aprovisionar las maquinas para los nodos del kluster, hay varias opciones, Azure, Amazon EKS, Amazon EC2, etc. Pero el que voy a elegir es Harvester. Aquí también está la opción de elegir que tipo de distribución de kubernetes vamos a instalar, las opciones que hay son RKE2, RKE1 y KS3. Yo voy a instalar RKE1 ya que es la versión más compatible con la instalación de rancher que tengo.



Una vez se elige la distribución de kubernetes y el origen del aprovisionamiento se abre la ventana de creación del clúster, aquí hay que configurar varias cosas, lo primero seria configurar varias plantillas de nodos, estas plantillas son definiciones de las características de un tipo de nodo. En este caso haré dos, una para los nodos "Master" y otra para los nodos "Workers". Para crear estas plantillas hay que pulsar en el botón de "Add Node Template" que aparece dentro de cada "Node pool", el cual es una definición de la cantidad de maquinas que cumplirán una función dentro del clúster, en este caso haré tres node pools distintas, una para las funciones de "Master", otra para las funciones de "Worker" y otra para las funciones de "Worker-etcd".



Solo voy a usar nodos con plantillas "Master" y "Workers" ya que no me hace falta más para el proyecto, pero en el caso de un entorno de producción de una empresa podríamos diferenciar otros tipos de nodos que sirvieran para cosas más específicas como por ejemplo nodos para aplicaciones con alta prioridad que tengan más recursos y más replicas, nodos para aplicaciones más pesadas que deban correr en una misma maquina, etc.

Una vez abro la creación de la plantilla del nodo, lo primero que hay que configurar es unas "Cloud Credentials", estas sirven para más tarde poder darle acceso a distintos usuarios a las maquinas del clúster, es simplemente crear una nueva o asignar una ya existente y luego la podre asignar a los usuarios.

Lo siguiente que hay que configurar son las características hardware del nodo, el namespace donde se van a alojar los nodos, el usuario ssh con el que se puede acceder al SO y los volúmenes de datos que tendrá ese nodo. Dentro del volumen root hay que elegir la imagen del sistema que he creado anteriormente.

The screenshot shows a user interface for creating a node template. It is divided into several sections:

- Account Access:** A section titled "Configure KubeconfigContent" with a dropdown menu labeled "KubeCred" and a "Add New" button.
- 2. Instance Options:** A section titled "Choose the size and OS of the virtual machine" with fields for:
 - CPUs *: 2
 - Cores: 8
 - Memory *: 8 GiB
 - Namespace *: kubecluster
 - SSH User *: rocky
- Volumes:** A section for configuring volumes:
 - Image Volume:** Fields for "Image *" (rocky-linux) and "bootOrder *" (1).
 - Disk *:** Field for "Disk" (40 GiB).
- Buttons:** At the bottom left are two buttons: "+ Add Volume" and "+ Add Image Volume".

Ahora hay que configurar las distintas redes a las que estará conectado el nodo, en este caso estará únicamente conectado a la red de VM (129).

Networks

Network	Remove
default/vm-network (vlanId=129)	

+ Add Network

Por último, hay que configurar un nombre para la plantilla y las etiquetas y taints que va a tener este nodo de kubernetes, en el master he añadido un taint que impide que se añada ningún pod nuevo a esa maquina que no este explícitamente especificado. Para terminar pulso el botón de "Create" de abajo y procedo a crear la plantilla del Worker.

RANCHER TEMPLATE

Name *: Master-Rocky Add a Description

Labels
Labels are key/value pairs that can be used to annotate containers and make scheduling decisions.

Key *	Value
Type	= Master -

ProTip: Paste one or more lines of key=value pairs into any key field for easy bulk entry.

+ Add Label

Taints
Taints mark that the node should not accept any pods that do not tolerate the taints.

Key *	Value	Effect
Master	True	NoSchedule -

Taint must be unique by key and effect pair.

+ Add Taint

Las diferencias entre la plantilla del master y worker son que el master tiene especificaciones hardware menores ya que este no va a alojar ningun pod que no sea del propio kubernetes y que en el worker no añado el taint.

Cloud Credentials

- KubeCred

2. Instance Options
Choose the size and OS of the virtual machine

CPU * 4	Memory * 32 GiB				
Namespace * kubecluster	SSH User * rocky				
Volumes					
Image Volume <table border="1"> <tr> <td>Image * rocky-linux (kubecluster/image-g4dlc)</td> <td>bootOrder * 1</td> </tr> <tr> <td>Disk * 100 GiB</td> <td></td> </tr> </table>		Image * rocky-linux (kubecluster/image-g4dlc)	bootOrder * 1	Disk * 100 GiB	
Image * rocky-linux (kubecluster/image-g4dlc)	bootOrder * 1				
Disk * 100 GiB					
+ Add Volume + Add Image Volume					

RANCHER TEMPLATE

Name *
Worker-Rocky

Add a Description

Labels
Labels are key/value pairs that can be used to annotate containers and make scheduling decisions.

Key *	Value
Type	= Worker

ProTip: Paste one or more lines of key=value pairs into any key field for easy bulk entry.
[+ Add Label](#)

Taints
Taints mark that the node should not accept any pods that do not tolerate the taints.

Engine Options
Customize the configuration of the Docker daemon

[Cancel](#) [Create](#)

Una vez configuradas las plantillas, tengo que elegir las funciones de cada tipo de maquina, hay 3 funciones que pueden cumplir los nodos, "etcd", el cual es un sistema de control de datos que debe tener 1,3 o 5 nodos como máximo, "Control Plane", esto hace referencia a los nodos que van a ser master del clúster y "Worker" que por el contrario serán los nodos que se dedicaran para la ejecución de las aplicaciones.

Las otras opciones que tengo son "Drain before delete" lo cual hace referencia a si cuando se vaya a borrar un nodo de ese tipo hay que esperar a que los pods que haya ejecutándose en el se trasladen a otro nodo o se destruyan junto a el, "Auto Replace", si ponemos aquí un tiempo específico sera el tiempo máximo que podrá estar un nodo sin responder antes de que se elimine automáticamente y por último se pueden poner taints específicos para cada pool a parte de los que tenga el node template.

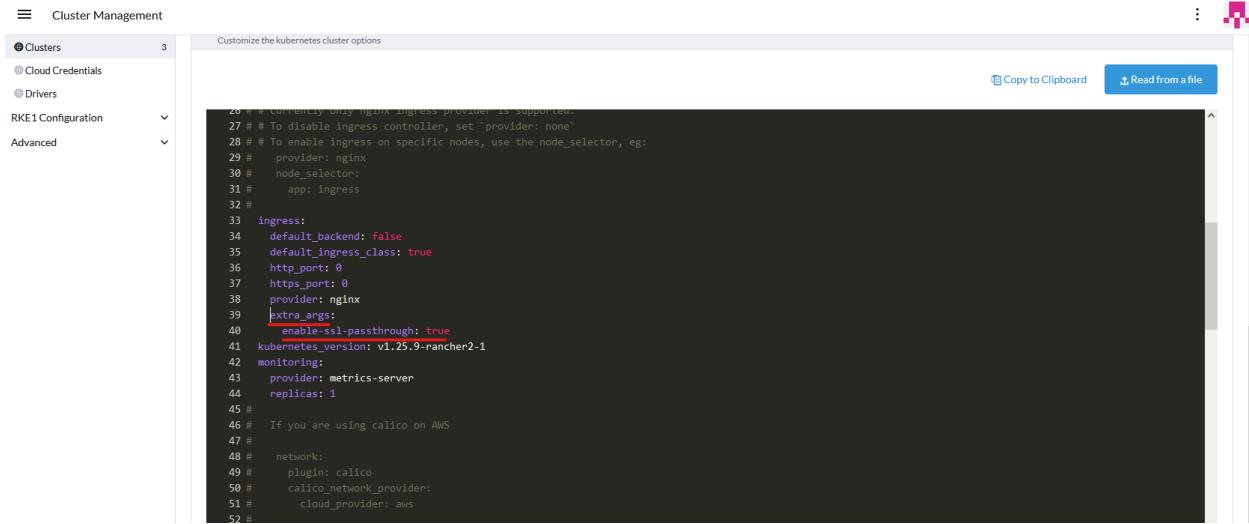
Name Prefix	Count	Template	Auto Replace	Drain Before Delete	etcd	Control Plane	Worker	Taints
Master	1	Master-Debian	0 minutes	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Worker-etcd	2	Worker-Debian	15 minutes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Worker	1	Worker-Debian	15 minutes	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Ahora, al igual que hice en el clúster de rancher, voy a activar la posibilidad de "ssl-pass-through", para ello, en la sección de "Cluster Options" pulso en el botón "Edit as YAML".

Haciendo esto, se abre un editor con el yaml que tiene todas las configuraciones de el clúster que se va a crear, para activar el ssl-pass-through hay que añadir las siguientes lineas debajo de la sección "ingress":

```
extra_args:
  enable-ssl-passthrough: true
```

Quedara de la siguiente forma:

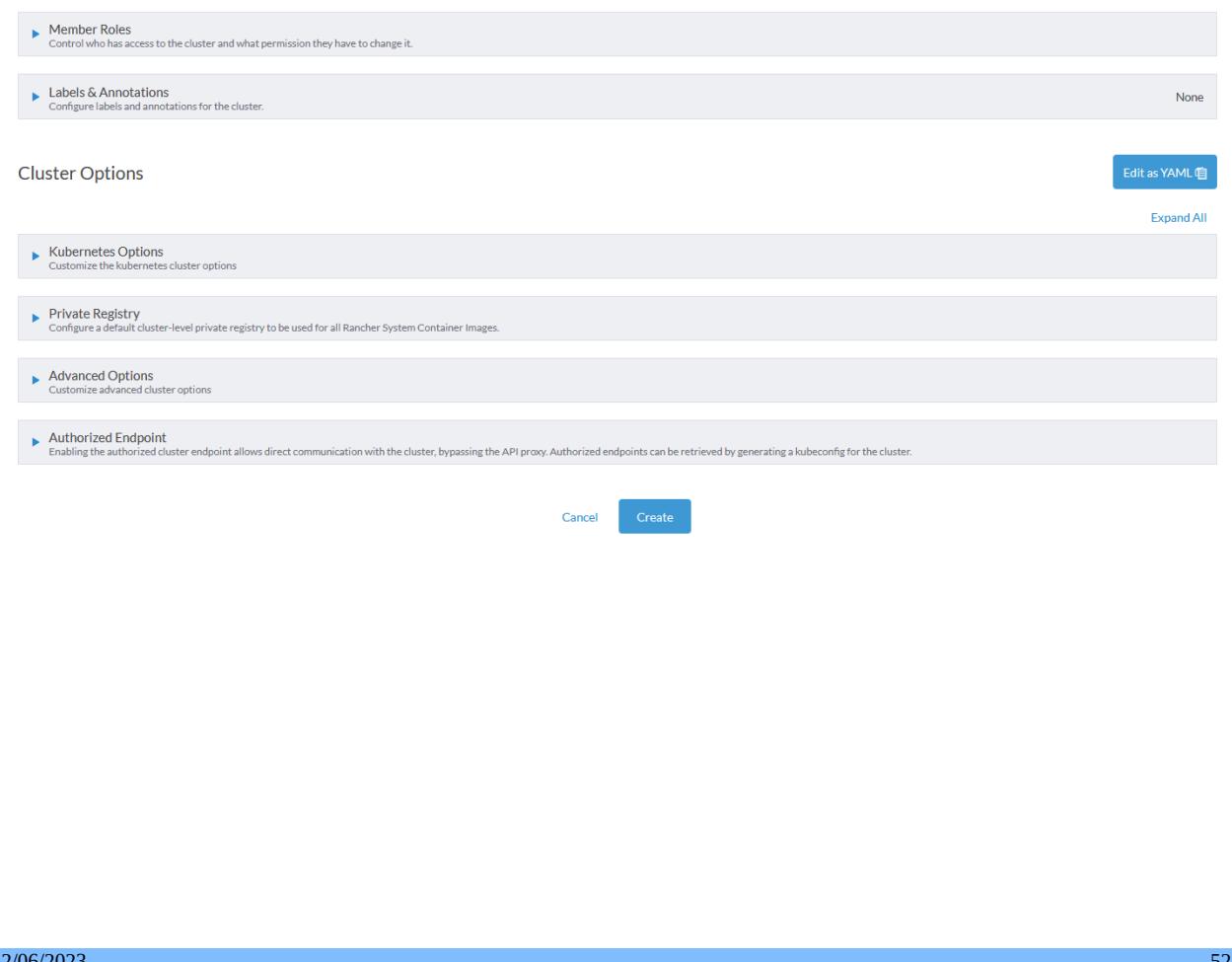


```

26 # = Currently only nginx ingress provider is supported.
27 # # To disable ingress controller, set `provider: none`
28 # # To enable ingress on specific nodes, use the node_selector, e.g:
29 #   provider: nginx
30 #   node_selector:
31 #     app: ingress
32 #
33   ingress:
34     default_backend: false
35     default_ingress_class: true
36     http_port: 0
37     https_port: 0
38     provider: nginx
39     extra_args:
40       enable-ssl-passthrough: true
41     kubernetes_version: v1.25.9-rancher2-1
42     monitoring:
43       provider: metrics-server
44     replicas: 1
45 #
46 # If you are using calico on AWS
47 #
48 #   network:
49 #     plugin: calico
50 #     calico_network_provider:
51 #       cloud_provider: aws
52 #

```

Hechas estas configuraciones, no haría falta tocar nada más a parte de darle un nombre al clúster que vamos a crear, pero en caso de ser necesario o querer especificar alguna configuración más avanzada del clúster, como los usuarios de rancher que tienen acceso a este clúster o la versión de RKE que vamos a instalar, se puede hacer con el resto de opciones de creación del clúster. En este clúster he dejado las opciones por defecto.



Cluster Options

[Edit as YAML](#) [Expand All](#)

- Kubernetes Options** Customize the kubernetes cluster options
- Private Registry** Configure a default cluster-level private registry to be used for all Rancher System Container Images.
- Advanced Options** Customize advanced cluster options
- Authorized Endpoint** Enabling the authorized cluster endpoint allows direct communication with the cluster, bypassing the API proxy. Authorized endpoints can be retrieved by generating a kubeconfig for the cluster.

[Cancel](#) [Create](#)

Por último, pulso el botón de "Create" y empezara la creación del clúster. Si pulso sobre el clúster en la interfaz de rancher se puede ir viendo el punto del proceso de creación en el que se encuentra el clúster o si ha habido algún tipo de error durante el mismo.

Clusters

Cluster: argo-cluster Provisioning

Namespaces: fleet-default Age: 6 secs

waiting for etcd, controlplane and worker nodes to be registered

Provisioner: RKE Machine Provider: Harvester Kubernetes Version: v1.25.9-rancher2-1 Machine Pools: 3 Machines: 4

Machine Pools Provisioning Log Snapshots Conditions Recent Events Related Resources

Pool: master Harvester - No Location / No Size (Master-Debian)

- Provisioning m-2fq8z - -/ - - Worker 12 secs
- (master1) Waiting for node become Running

Pool: worker Harvester - No Location / No Size (Worker-Debian)

- Provisioning m-hjs9k - -/ - - Worker 12 secs
- (worker1) Waiting for node become Running

Pool: worker-etcd Harvester - No Location / No Size (Worker-Debian)

- Provisioning m-g9g6g - -/ - - Worker, Etcd 12 secs
- (worker-etcd1) Waiting for node become Running
- Provisioning m-pnr55 - -/ - - Worker, Etcd 12 secs
- (worker-etcd2) Waiting for node become Running

Si voy a el dashboard de Harvester y miro la sección de "Virtual Machines" veré que se han creado automáticamente las maquinas de los nodos.

Virtual Machines

Namespace: kubecluster

State	Name	CPU	Memory	IP Address	Node	Age
Running	master1	2	8 Gi	molden-harvester-03	40 secs	
Running	worker-etcd1	4	32 Gi	molden-harvester-01	40 secs	
Running	worker-etcd2	4	32 Gi	molden-harvester-02	40 secs	
Starting	worker1	4	32 Gi	molden-harvester-02	40 secs	

Guest VM is not reported as running

Una vez el clúster este creado y en funcionamiento, pasara a el estado activo.

The screenshot shows the 'Cluster Management' interface for a cluster named 'argo-cluster' (Active). The cluster was created 12 mins ago using RKE as the provider, Harvester as the machine provider, and Kubernetes Version v1.25.9. It contains 3 machine pools and 4 machines. The 'Machine Pools' tab is selected, showing the following details:

Pool	Name	Node	External/Internal IP	OS	Roles	Age
Pool: worker	m-hjs9k	worker1	10.200.129.168	Linux	Worker	12 mins
Pool: master	m-2fq8z	master1	10.200.129.170	Linux	Control Plane, Etcd	12 mins
Pool: worker-etcd	m-g9g6g	worker-etcd1	10.200.129.169	Linux	Worker, Etcd	12 mins
	m-pnr55	worker-etcd2	10.200.129.172	Linux	Worker, Etcd	12 mins

Si quisiese escalar o desescalar el clúster, se puede hacer fácilmente desde esta misma pagina de "Cluster Management", simplemente pulsando en los botones de añadir o quitar maquinas de cada pool, aunque siempre teniendo en cuenta que debe haber 1, 3 o 5 maquinas con el rol etcd.

The screenshot shows the 'Cluster Management' interface for the same cluster, but with some nodes in a 'Provisioning' state. The 'Machine Pools' tab is selected, showing the following details:

Pool	Name	Node	External/Internal IP	OS	Roles	Age
Pool: worker	m-hfmd5	-	-/-	-	Worker	12 secs
	m-hjs9k	worker1	10.200.129.168	Linux	Worker	16 mins
Pool: master	m-2fq8z	master1	10.200.129.170	Linux	Control Plane, Etcd	16 mins
	m-jmcls	-	-/-	-	Control Plane, Etcd	18 secs
Pool: worker-etcd	m-g9g6g	worker-etcd1	10.200.129.169	Linux	Worker, Etcd	16 mins
	m-pnr55	worker-etcd2	10.200.129.172	Linux	Worker, Etcd	16 mins

The screenshot shows the Rancher UI for managing clusters. On the left, a sidebar menu includes 'Clusters' (3), 'Cloud Credentials', 'Drivers', 'RKE1 Configuration', and 'Advanced'. The main area displays the 'Cluster: argo-cluster (Active)' details, including the namespace 'fleet-default' and age '18 mins'. It lists the provisioner as 'RKE' and machine provider as 'Harvester'. The Kubernetes version is v1.25.9, with 3 machine pools and 5 machines. A table shows the machine pools: 'worker' (2 nodes: worker2 and worker1), 'master' (2 nodes: master1 and master2), and 'worker-etcd' (1 node: worker-etcd1). Each row includes columns for State, Name, Node, External/Internal IP, OS, Roles, and Age.

v2.7.3

Administrar el nuevo clúster

- Una vez esta creado el clúster, podemos acceder a su pantalla de administración desde el dashboard principal de rancher o desde el menú expansible de la izquierda.

The screenshot shows the Rancher main dashboard. At the top, it says 'Welcome to Rancher'. Below is a 'Clusters' section with two entries: 'argo-cluster' (Active, Harvester RKE, v1.25.9) and 'local' (Imported, Imported, v1.25.9). The 'Clusters' table has columns for State, Name, Provider, Kubernetes Version, CPU, Memory, and Pods. To the right is a 'Links' sidebar with links to 'Docs', 'Forums', 'Slack', 'File an Issue', 'Get Started', and 'Commercial Support'. At the bottom, there's a 'Home' button, an 'EXPLORE CLUSTER' section with 'argo-cluster' and 'local' buttons, and a 'GLOBAL APPS' section.

Si entro en la pagina del clúster puedo ver toda su infotmación, hosts, carga de cpu y ram, deployments, pods, etc.

Ademas de esto, arriba a la derecha hay varias opciones para administrar el clúster, se puede abrir directamente una terminal con kubectl, copiar o descargar el fichero de kubeconfig, importar archivos yaml al clúster o usar el buscador de recursos para encontrarlos de manera rápida. Ahí también se puede elegir que namespaces quiero ver, de los cuales solo aparecerán los que tenga acceso el usuario que este loggeado en ese momento.

Hay que tener en cuenta que solo se podrá acceder a kubectl desde rancher o con el archivo de kubeconfig en un equipo que este conectado a la misma red que el clúster

Ya tengo el clúster creado y funcionando, ahora podría empezar a desplegar aplicaciones a través de deployments y servicios como con cualquier otro clúster de kubernetes.

Manuales de Usuario

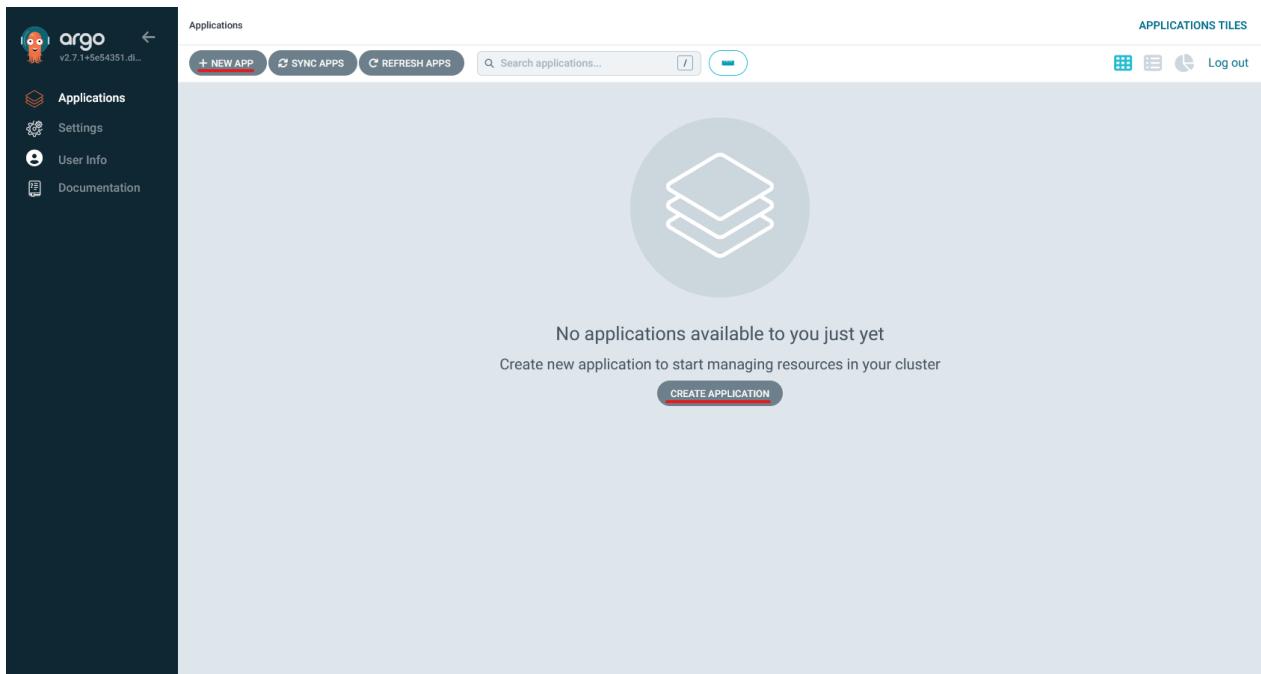
- En los siguientes manuales de “Usuario” voy a describir los distintos procesos que hay que seguir para, una vez instalado y configurado todo, desplegar y publicar en internet una aplicación.

• Despliegue de aplicación en ArgoCD

- Una vez tengo el clúster con argocd instalado ya puedo desplegar aplicaciones desde helm o github y programarlas para que se actualicen automáticamente cuando haya una nueva versión.

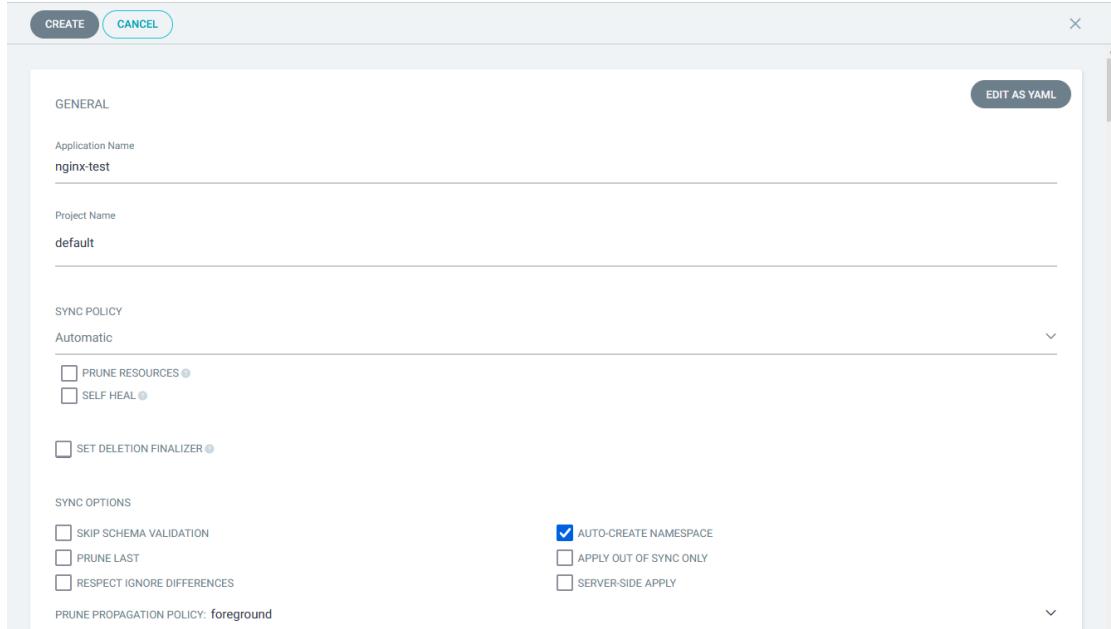
Yo voy a usar un ejemplo para github de una pagina web simple creada por el equipo argocd y uno para helm de una instalación de nginx básica, aunque la diferencia entre uno y otro es mínima.

Lo primero para crear una app en argocd es ir a la interfaz web (Aunque se puede hacer a través de comandos) y pulsar en el botón de "New App".



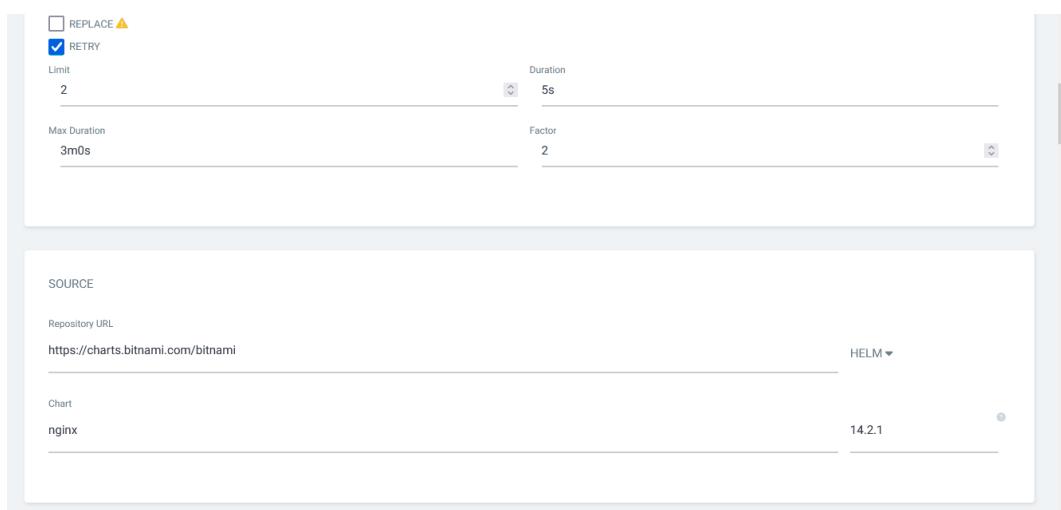
Se abre una ventana donde puedo configurar todas las opciones del despliegue de la aplicación, argocd tiene muchísimas opciones a la hora de como desplegar las aplicaciones, para estas dos aplicaciones de ejemplo lo único que voy a cambiar es el modo de sincronización para que sea automático.

Aquí también hay que elegir obligatoriamente un nombre para la aplicación y un proyecto donde estará la misma, este "Project" sera algo parecido a los namespaces de kubernetes y se pueden crear más desde el menú de opciones de ArgoCD, pero para estos ejemplos voy a usar el proyecto default que viene ya creado en rancher.



En la sección "Source" es donde hay que indicar el repositorio desde donde se va a sincronizar la aplicación, aquí es donde se elige entre un repositorio de helm o de git, sin embargo, para que argocd lo detecte, el repositorio de git debe tener la misma estructura que uno de helm.

Aquí hay que indicar la URL con el repositorio, en el caso de helm indicare el "chart" que se va a instalar, es decir, la aplicación, ya que en un mismo repositorio puede haber varias apps, y la versión de esta. Para github indicare la rama de git donde se encuentra la app y el path a la misma, aquí también se puede tener varias aplicaciones separadas en distintas carpetas.



The screenshot shows the 'SOURCE' configuration section of the Argocd UI. It includes fields for Repository URL (https://github.com/argoproj/argocd-example-apps), Revision (HEAD), and Path (helm-guestbook). There are also dropdowns for GIT and Branches.

Por último, hay que elegir el clúster en el que desplegar la app, que en este caso sera el clúster por defecto que es donde esta instalado argocd, ya que no he añadido ninguno más, y el namespace donde estará la app, al haber activado la opción de "Auto-Create Namespace" si no existe el namespace que ponga se creara automáticamente.

Las últimas opciones son opciones de personalización de cada aplicación creadas por el autor de la app, en la app de ejemplo de argocd no tengo que cambiar nada, pero en la de nginx tengo que cambiar el tipo de servicio a ClusterIP para poder reenviar el trafico desde haproxy. Es recomendable leer y configurar todas las opciones para adecuarlas a el entorno en el que se van a usar las aplicaciones.

The screenshot shows the 'DESTINATION' configuration section with a Cluster URL of https://kubernetes.default.svc and a Namespace of nginx-test. Below this, the 'HELM' configuration section is shown, featuring tabs for HELM, VALUES FILES, and VALUES. Under PARAMETERS, there are three entries: service.type (ClusterIP), autoscaling.enabled (false), and autoscaling.maxReplicas (with a 'Remove override' link).

Al pulsar "Create" arriba a la izquierda, se empieza a desplegar la app y, si todo esta correcto, en poco tiempo estará desplegada, a partir de ahora cualquier cambio efectuado en los repositorios conectados se verán reflejados en la app que tengo desplegada en mi clúster. ArgoCD comprobara automáticamente cada cierto tiempo si ha habido cambios y los sincronizara automáticamente, también esta la opción de sincronizar manualmente.

The screenshot shows the Argo CD interface with two applications listed:

- guestbook**: Project: default, Status: Healthy Synced, Repository: https://github.com/argoproj/argocd-example-apps, Target Revision: HEAD, Path: helm-guestbook, Destination: in-cluster, Namespace: guestbook, Created At: 05/17/2023 12:52:01 (a minute ago).
- nginx-test**: Project: default, Status: Healthy Synced, Repository: https://charts.bitnami.com/bitnami, Target Revision: 14.2.1, Chart: nginx, Destination: in-cluster, Namespace: nginx-test, Created At: 05/17/2023 12:43:46 (9 minutes ago).

The sidebar includes filters for SYNC STATUS (Unknown: 0, Synced: 2, OutOfSync: 0), HEALTH STATUS (Unknown: 0, Progressing: 0, Suspended: 0, Healthy: 2, Degraded: 0, Missing: 0), and LABELS.

Si pulso encima de cada aplicación, puedo ver todos los datos del despliegue.

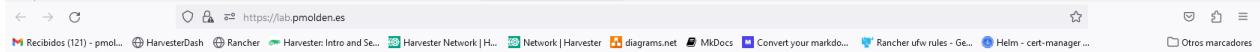
The screenshot shows the Argo CD interface for the guestbook application, which is healthy and synced to HEAD (53e28ff). The sync status is Sync OK, last sync was a few seconds ago (Wed May 17 2023 12:52:03 GMT+0200), and the author was May Zhang <may_zhang@intuit.com> with a comment: feat: update helm samples to use helm3 (#78).

The application details tree shows the deployment history:

- guestbook (Synced)
- guestbook-helm-guestbook (Synced)
- guestbook-helm-guestbook-6... (Synced)
- guestbook-helm-guestbook-j8... (Synced)
- guestbook-helm-guestbook-ep (Synced)
- guestbook-helm-guestbook-rs (Synced)
- guestbook-helm-guestbook-pod (Running)

The sidebar includes filters for NAME, KINDS, and SYNC STATUS (Synced: 2, OutOfSync: 0).

Ya tengo las aplicaciones desplegadas, ahora solo faltaría configurar el acceso a ellas con Ingress y haproxy y ya podre acceder a ellas a través del navegador desde cualquier sitio.

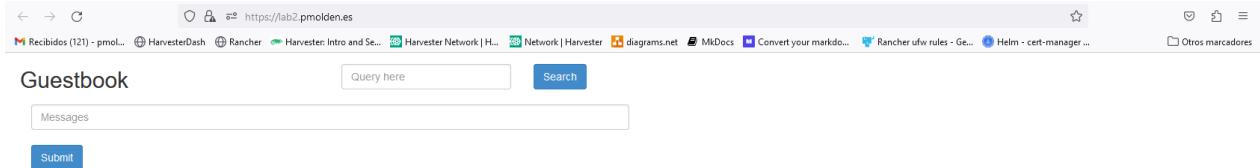


Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.



• Configurar Ingress

- Ahora tengo que configurar el ingress tanto en el clúster local como en el clúster de argocd para reenviar el trafico dirigido a las distintas aplicaciones, voy a configurar ingress para tener acceso desde fuera de la red de la empresa a la aplicación web de argocd y también he desplegado un chart de nginx con helm para usarlo de ejemplo.

Cluster local

- En el clúster local voy a crear dos reglas de ingress en el namespace de haproxy en el menú de "Service Discovery -> Ingresses", una para argocd y otra para nginx, en ambos casos tengo que poner como host el subdominio DNS que les vaya a asignar, los cuales deben apuntar ambos a la ip publica del servidor de rancher.

Para argocd voy a usar el subdominio argocd.pmolden.es y para nginx lab.pmolden.es, en ambos el tipo de "Path" debe ser "ImplementationEspecific" y la "Ingress Class" debe ser nginx.

En argocd el ingress debe apuntar al servicio haproxy en el puerto 443 y en el de nginx al puerto 80.

Por último, solo en el ingress de argocd, en la pestaña de "Labels & Annotations" debo poner una nueva annotation para activar el "ssl-passthrough", esto sirve para que ingress pase directamente todos los datos de la conexión ssl sin modificarlos ni redirigirlos ya que por defecto sería ingress el que maneja los certificados y hace redirecciones automáticamente del puerto 80 al 443, pero para argocd, que tiene su propia redirección y certificado, no funcionaría y entraría en un bucle de redireccionamiento.

nginx.ingress.kubernetes.io/ssl-passthrough=true

The screenshot shows the Argo CD interface for managing Kubernetes resources. On the left, a sidebar lists various clusters, workloads, and resources. The main panel shows an 'Ingress: argocd' resource under the 'haproxy' namespace. The 'Labels & Annotations' tab is active, displaying a table with one entry:

Key	Value
nginx.ingress.kubernetes.io/ssl-passthrough	true

Below the table is a 'Remove' button and a 'Add Annotation' button.

Para nginx, como no tiene su propio redireccionamiento a https, se puede añadir aquí un certificado ssl para que lo gestione ingress, en este caso voy a usar el certificado autofirmado que trae por defecto, pero podría generar un certificado valido con letsencrypt, incluso de manera automática con cert-manager.

The screenshot shows the Argo CD interface for managing Kubernetes resources. On the left, a sidebar lists various clusters, workloads, and resources. The main panel shows an 'Ingress: nginx-test' resource under the 'haproxy' namespace. The 'Certificates' tab is active, displaying a table with one entry:

Certificate - Secret Name*	Host
Default Ingress Controller Certificate	lab.pmolden.es

Below the table is a 'Remove' button and a 'Add Host' button.

Cluster ArgoCD

- En el clúster de argocd debo crear también las mismas reglas de ingress que en el local, pero esta vez cada uno en el namespace de su aplicación y apuntando a su servicio correspondiente.

Al igual que antes, en el ingress de argocd tengo que poner la anotación del "ssl-passthrough".

The screenshot shows the ArgoCD interface for managing Kubernetes resources. On the left, there's a sidebar with navigation links like Cluster, Workloads, Apps, etc. The main area is focused on an 'Ingress' resource named 'argocd' in the 'argocd' namespace. The 'Rules' tab is selected, displaying a configuration for a specific host and path mapping to a target service and port.

This screenshot shows the ArgoCD interface for managing Kubernetes resources. It displays an 'Ingress' resource named 'nginx-test' in the 'nginx-test' namespace. The 'Rules' tab is active, showing a configuration where traffic to 'lab' is directed to the 'nginx-test' service on port 80 via the path '/foo'.

Una vez creados, puedo acceder sin problema a los servicios de nginx y argocd desde dentro o fuera de la red de la empresa con los nombres dns que le he asignado.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

• Certificado Letsencrypt

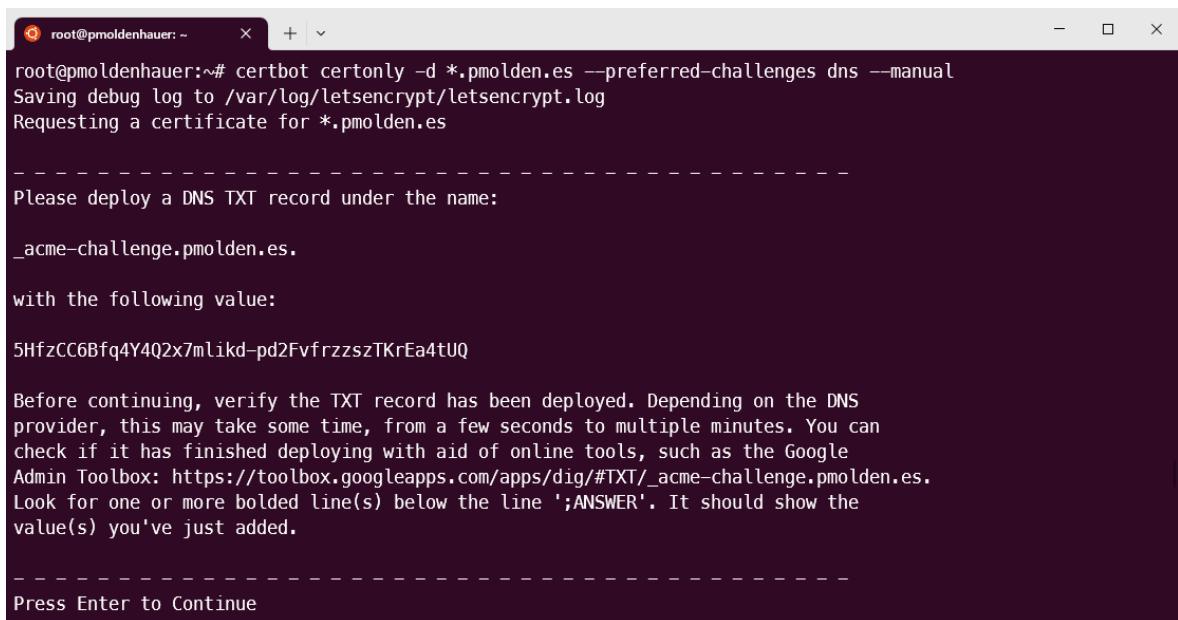
- Para poder securizar las aplicaciones que despliego con tls he decidido hacer un certificado wildcard (*.pmolden.es) con letsencrypt. Para ello he seguido los siguientes pasos:

1. Instalo certbot en el servidor de rancher que es donde esta asignada la ip publica.

```
sudo snap install core; sudo snap refresh core
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

2. Ejecuto cert bot con los parámetros para hacer una petición de certificado tls de manera manual por el método de registro txt en el dns.

```
certbot certonly -d *.pmolden.es --preferred-challenges dns --manual
```



A terminal window titled 'root@pmoldenhauer: ~' showing the execution of the 'certbot certonly' command. The command is run with parameters: '-d *.pmolden.es', '--preferred-challenges dns', and '--manual'. The output indicates that a certificate is being requested for the domain *.pmolden.es. It prompts the user to add a DNS TXT record under the name '_acme-challenge.pmolden.es' with the value '5HfzCC6Bfq4Y4Q2x7mlkd-pd2FvfrzzszTKrEa4tUQ'. A note is provided about verifying the deployment of the TXT record, mentioning the Google Admin Toolbox for checking. Finally, it prompts the user to press Enter to continue.

```
root@pmoldenhauer:~# certbot certonly -d *.pmolden.es --preferred-challenges dns --manual
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Requesting a certificate for *.pmolden.es

-----
Please deploy a DNS TXT record under the name:

_acme-challenge.pmolden.es.

with the following value:

5HfzCC6Bfq4Y4Q2x7mlkd-pd2FvfrzzszTKrEa4tUQ

Before continuing, verify the TXT record has been deployed. Depending on the DNS provider, this may take some time, from a few seconds to multiple minutes. You can check if it has finished deploying with aid of online tools, such as the Google Admin Toolbox: https://toolbox.googleapps.com/apps/dig/#TXT/_acme-challenge.pmolden.es.
Look for one or more bolded line(s) below the line ';ANSWER'. It should show the value(s) you've just added.

-----
Press Enter to Continue
```

3. Por último, sigo las instrucciones que aparecen en la terminal, hago un nuevo registro de tipo TXT en el dominio pmolden.es con el nombre y contenido que me indica, una vez hecho esto continuo el proceso pulsando enter y ya tendré disponible mi certificado con su llave privada en /etc/letsencrypt/live/pmolden.es

```
Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/pmolden.es/fullchain.pem
Key is saved at:      /etc/letsencrypt/live/pmolden.es/privkey.pem
This certificate expires on 2023-08-29.
These files will be updated when the certificate renews.

NEXT STEPS:
- This certificate will not be renewed automatically. Autorenewal of --manual certificates requires the use
of an authentication hook script (--manual-auth-hook) but one was not provided. To renew this certificate, r
epeat this same certbot command before the certificate's expiry date.

-----
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:  https://letsencrypt.org/donate
 * Donating to EFF:                  https://eff.org/donate-le
-----
```

Ahora puedo usar ese certificado para proteger mis aplicaciones, ya sea usando el certificado a través de ingress con un secret de kubernetes o de la propia app. Los certificados de letsencrypt duran tres meses, para renovarlo, lo puedo hacer de manera manual o crear un script para ello.

Conclusiones finales

- Como conclusión al sistema y proyecto realizados, me parece que, a pesar de tener cierta dificultad a la hora de entender por primera vez todos los componentes, una vez que los entiendes, es fácil instalar y configurar todo. Ademas, es un sistema sólido y bastante a prueba de fallos, que usa tecnología moderna, opensource y dirigida a empresas, con lo cual, es muy fiable. Teniendo todo esto en cuenta, me parece una solución muy viable para empresas medianas-grandes que necesiten de un sistema de virtualización de alta disponibilidad.

Con respecto al proyecto en si, me parece que he alcanzado el punto en el que se podría empezar a utilizar para desarrollo o testeo, pero aun habría muchas cosas que mejorar, como por ejemplo el sistema de creación de backups, una monitorización central, escalado de las aplicaciones de manera automática según demanda, etc. Aunque, todo esto que me ha faltado, ha sido porque he considerado que no era necesario configurarlo para este proyecto por su simplicidad a la hora de hacerlo si fuese necesario o, en algunos casos como los backups, por falta de más recursos que serían excesivos para un proyecto de este estilo o por falta de tiempo.

Dicho todo esto, personalmente estoy muy contento con el resultado de mi proyecto y con la documentación que he elaborado del mismo, me ha servido para aprender muchas tecnologías y conceptos nuevos y reafirmar algunos que ya había aprendido.

Para la realización de este documento he intentado resumir toda la documentación que he ido recopilando durante el proyecto, la documentación completa se encuentra en:
<https://github.com/firex20/ProyectoFinalCiclo>

Grado de cumplimiento de los objetivos fijados

- Creo que he cumplido todos los objetivos que me había marcado al principio del proyecto, aunque por supuesto, como he dicho en la conclusión final, con más tiempo y recursos este sistema se podría mejorar mucho, algunas de las cosas que se me ocurren para mejorar el proyecto podrían ser por ejemplo:

- Configurar los backups de las distintas aplicaciones/clusteres/maquinas virtuales de las que hiciesen falta
- Configurar un sistema de monitorización centralizado, donde se pueda consultar todos los datos tanto de los host físicos como de los virtuales y de las aplicaciones
- Configurar el auto-escalado de las aplicaciones basado en la demanda
- Separar el sistema de balanceo de cargas (Haproxy) de el servidor de rancher y situarlo en un nuevo clúster con alta disponibilidad, pudiendo así eliminar uno de los saltos en el reenvío de la conexión a las aplicaciones.
- Automatizar la renovación de los certificados con cert-manager.

Con estas cosas, creo que el proyecto ya estaría en el punto en el que se podría poner en producción para ofrecer servicios a distintos clientes.

Bibliografía empleada.

Instalacion-Configuración Harvester

- <https://docs.harvesterhci.io/v1.1>
- <https://www.balena.io/etcher>
- <https://docs.harvesterhci.io/v1.1/advanced/storagenetwork/>
- <https://github.com/harvester/harvester/issues/2586>

Instalación Rancher

- <https://rockylinux.org/>
- <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-rocky-linux-9>
- <https://docs.docker.com/engine/install/centos/>
- <https://www.tecmint.com/open-port-for-specific-ip-address-in-firewalld/>
- <https://www.rancher.com/products/rke>
- <https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>
- <https://helm.sh/docs/intro/install/>
- <https://cert-manager.io/docs/installation/helm/>
- <https://www.rancher.com/>
- <https://ranchermanager.docs.rancher.com>

Cluster Kubernetes

- https://dl.rockylinux.org/vault/rocky/9.0/images/x86_64/
- <https://cloud.debian.org/images/cloud/bullseye/latest/>
- <https://www.reddit.com/r/rancher/comments/y3uvmg/comment/isp9zmb/>

ArgoCD

- <https://argo-cd.readthedocs.io/en/stable/>
- https://argo-cd.readthedocs.io/en/stable/getting_started/
- <https://kubernetes.io/docs/tasks/administer-cluster/dns-debugging-resolution/>
- <https://devpress.csdn.net/k8s/62fcc33e7e66823466190886.html>
- <https://github.com/argoproj/argocd-example-apps>
- <https://artifacthub.io/packages/helm/bitnami/nginx>
- <https://github.com/gd4Ark/star-battle>
- <https://computingforgeeks.com/perform-git-clone-in-kubernetes-pod-deployment/>
- <https://github.com/firex20/helm-example-app>
- <https://artifacthub.io/packages/helm/nextcloud/nextcloud>

HaProxy

- <https://www.haproxy.com/blog/haproxy-configuration-basics-load-balance-your-servers/>
- https://hub.docker.com/_/haproxy
- <https://ranchermanager.docs.rancher.com/how-to-guides/new-user-guides/kubernetes-resources-setup/encrypt-http-communication>
- <https://www.suse.com/support/kb/doc/?id=000020147>
- <https://serversforhackers.com/c/using-ssl-certificates-with-haproxy>
- <https://rancher.support/training/rke/rke-cluster-yaml-breakdown/#ingress>
- <https://www.haproxy.com/blog/how-to-map-domain-names-to-backend-server-pools-with-haproxy>

Ansible

- <https://fabianlee.org/2021/05/29/ansible-orchestrating-ssh-access-through-a-bastion-host/>
- <https://docs.ansible.com/>
- <https://ansible-lint.readthedocs.io/configuring/>
- <https://www.redhat.com/sysadmin/ansible-templates-configuration>
- <https://www.buggycoder.com/ansible-error-failed-to-set-permissions/>

Letsencrypt

- <https://certbot.eff.org/instructions>
- <https://dev.to/devlix-blog/automate-let-s-encrypt-automate-let-s-encrypt-wildcard-certificate-creation-with-ionos-dns-rest-api-o23>
- <https://eff-certbot.readthedocs.io/en/stable/using.html#manual>