





CI/CD con Gitlab en Rancher + K3s

GitLab K3S

Jose Carlos Rodríguez Cañas

IES Gonzalo Nazareno





Índice

1.	Introducción	. 3
	1.1. Objetivos que se quieren conseguir	3
	1.2. ¿Por qué esta estructura y es necesario Rancher?	4
2.	Fundamentos y conceptos básicos	. 5
	2.1. K8s + K3s como clúster	5
	2.2. ¿Qué es Rancher? Funcionamiento y compatibilidades	7
	2.3. Grafana + Prometheus para monitoreo y observabilidad	8
	2.4. Gitlab + Gitlab Runner como CI/CD	10
3.	Escenario que se ha realizado	12
	3.1. Tipos de estructuras en Rancher	12
	3.2. Escenario elegido con todas sus características	16
	3.3. Instalaciones	17
	3.3.1. Nodos mediante KVM	17
	3.3.2. Rancher y cluster local de K3s	19
	3.3.3. Grafana + Prometheus	28
	3.3.4. Gitlab y Gitlab Runner	31
	3.4. Configuraciones	38
	3.4.1. Configuración del clúster	38
	3.4.2. Configuración de métricas con Grafana + Prometheus	43
	3.4.3. Despliegue y configuración del Runner en Gitlab	49
4.	Demostración final	58
5.	Dificultades encontradas	64
	5.1. Conexión del cluster con Rancher	64
	5.2. Configuración de Gitlab y su respectivo Runner	65
6.	Conclusión	67
7.	Bibliografía	69







1. Introducción

Esta memoria constituye una visión detallada del proyecto integrado, el cual está dirigido hacia **la creación de una plataforma** robusta de **gestión de clústeres de contenedores de código abierto**. En este proyecto se abordarán los aspectos esenciales relacionados con el **despliegue continuo, monitorización, implementación y creación de un cluster efectivo de producción con Rancher y K3s**.

1.1. Objetivos que se quieren conseguir

Los objetivos principales de este proyecto son los siguientes:

- Implementación de un Cluster de K3s y Rancher: El objetivo es entender sobre el despliegue de un cluster de K3s complementado con un orquestador de aplicaciones y contenedores como Rancher.
- Investigación y Selección de Tecnologías Relevantes: Ésto implica identificar herramientas, frameworks y plataformas que optimicen el despliegue, monitoreo y seguridad del clúster de contenedores (implementado en las métricas de Grafana y Prometheus), manteniéndolo en consonancia con las mejores prácticas de la industria.
- Priorización de la Disponibilidad y Eficiencia: La importancia de garantizar la disponibilidad y eficiencia tanto para los administradores de sistemas como para los desarrolladores. Ésto nos servirá para la adopción de prácticas y herramientas que faciliten la gestión del clúster, minimizando tiempos de inactividad y optimizando recursos.
- Desarrollo de Herramientas de Supervisión y Seguridad: Ésto implica la implementación de soluciones de monitoreo continuo, detección de amenazas y políticas de acceso, que garanticen la integridad y confidencialidad de los datos alojados en el clúster.





- Facilitar la Implementación para Desarrolladores: Ésto se logrará mediante la creación de herramientas y procesos automatizados que simplifiquen el despliegue y actualización de aplicaciones en el clúster.
- Promoción de la Eficiencia y Seguridad en el Desarrollo y **Desplieque de Aplicaciones:** Finalmente, se aspira a promover la eficiencia y seguridad en el ciclo de vida completo de las aplicaciones, desde su desarrollo hasta su despliegue en entornos de producción de manera continua.

<u>1.2. ¿Por qué esta estructura y es necesario Rancher?</u>

En esta estructura se basa en crear un entorno de desarrollo y despliegue de aplicaciones que sea robusto, escalable y fácil de gestionar para el administrador de sistemas. Por ello, utilizando herramientas como Gitlab, Rancher, K3s... Buscamos establecer un ambiente que permita la administración eficiente de contenedores y la implementación continua de aplicaciones.

Rancher es una pieza clave en nuestra estructura por varias razones. Lo más importante es que nos facilita la administración del clúster de contenedores, proporcionando una interfaz gráfica intuitiva y funcionalidades avanzadas para la configuración, monitoreo y escalado de aplicaciones. Ésto nos ayuda a la gestión del clúster, especialmente en entornos complejos con múltiples nodos y servicios.

Además, Rancher ofrece características como el catálogo de aplicaciones, que permite desplegar aplicaciones preconfiguradas con solo unos pocos clics. También proporciona herramientas para el registro y la autenticación de usuarios, lo que garantiza un acceso seguro al clúster y sus recursos.

En resumen, la estructura propuesta con Rancher es totalmente necesaria porque proporciona una plataforma que nos permite integrar de manera fácil de usar para la administración de clústeres de contenedores.





2. Fundamentos y conceptos básicos

Para poder proceder al montaje del entorno de producción, tendremos que entender cada uno de los servicios y aplicaciones que vamos a instalar en nuestro clúster.

2.1. K8s + K3s como clúster

Antes de comenzar las explicación de las tecnologías, tenemos que explicar que Kubernetes y K3s ya que son las bases de este proyecto.

Kubernetes, o más bien conocido como K8s, es una plataforma portable y extensible de código abierto que nos permite administrar las cargas de trabajo y servicios en contenedores. Además, Kubernetes facilita la automatización y permite la configuración declarativa.

K8s se puede entender como:

- Una plataforma de contenedores
- Una plataforma de microservicios
- Una plataforma portable de nube

y de muchas más maneras.



Con esto comentado, podemos ver que Kubernetes es una plataforma más que válida para la orquestación de clústers y contenedores





pero... y ¿si llegamos más lejos? Por ello, implementamos K3s con la ayuda de Rancher (lo comentaremos en el siguiente apartado). K3s es una distribución de Kubernetes certificada y de alta disponibilidad diseñada para cargas de trabajo de producción en ubicaciones remotas, desatendidas y con recursos limitados o dentro de dispositivos IoT.

Una de las ventajas de utilizar K3s es que está empaquetado en un único binario (reconocido como Single binary), el cuál tiene un almacenamiento en menos de 70 MB de almacenamiento. Ésto reduce las dependencias y los pasos necesarios para instalar, ejecutar y actualizar un clúster de Kubernetes de producción.



Un dato curioso de K3s es cómo está optimizado tanto para ARM64 como para ARMv7, son compatibles con los binarios e imágenes de múltiples arquitecturas. Por ello, trabaja muy bien con algo pequeño como una Raspberry PI o como una instancia en AWS con un almacenamiento mínimo de 32GiB.

Como hemos comentado anteriormente, tendremos que integrar K3s con Rancher pero... ¿Qué es rancher y para que se utiliza? Lo veremos en el siguiente apartado.







2.2. ¿Qué es Rancher? Funcionamiento y compatibilidades

Rancher es un software de administración de contenedores de **código abierto** que nos sirve para gestionar aplicaciones de contenedores en entornos virtuales. Además, está perfectamente diseñado para integrarlo con **Kubernetes** o por ende, en nuestro caso utilizaremos **K3s**.

Aborda los desafíos operativos y de seguridad de administrar múltiples clústeres de **Kubernetes**, al tiempo que proporciona a los equipos de **DevOps** herramientas integradas para ejecutar cargas de trabajo en contenedores.

Otras características relevantes son la posibilidad de orquestación de servicios de almacenamiento persistente para **Docker**, compatibilidad con **Docker Machine** y la creación e importación de **clusters** como **Amazon EKS, Azure AKS y Google GKE**.



Por último, tenemos que entender que es el **Edge Computing** es el esquema que vamos a montar en Rancher. Este proporciona K3 para poder soportar la ejecución de Kubernetes dentro de los nodos edge.





2.3. Grafana + Prometheus para monitoreo y observabilidad

Antes de sumergirnos en el entorno de integración continua como Gitlab, es esencial comprender el funcionamiento de herramientas de monitoreo y observabilidad como Grafana y Prometheus.

Grafana es una herramienta poderosa para visualizar datos de series temporales, con una amplia integración con diversas fuentes de datos como InfluxDB, PostgreSQL, entre otras.

Además, su comunidad activa proporciona información valiosa y una variedad de plugins que permiten conectar con distintos sistemas. También ofrece la capacidad de enviar alertas de manera rápida y sencilla a través de correo electrónico.

Por otro lado, Prometheus es un software especializado en monitoreo y alertas, escrito en el lenguaje de programación Go. Éste recopila métricas de infraestructuras y aplicaciones, tales como datos sobre la CPU, RAM, rendimiento, aplicaciones en **Docker** o **K8s**, entre otros, almacenandose en su base de datos en forma de series temporales.

Algunas de sus características principales incluyen un modelo de datos multidimensional, un lenguaje de consultas propio, nodos autónomos de servidor único y configuración estática.





Una pregunta común que surge es cómo conectar Grafana y Prometheus para una monitorización y observabilidad eficientes. Ésto se logra mediante métricas, para lo cual Prometheus ofrece una herramienta llamada **node exporters**. Éstos pueden ser scripts o servicios que recolectan **métricas** específicas del sistema y las proporcionan en un formato adecuado para Prometheus.

Los **node exporters** son esenciales para el funcionamiento de Prometheus, ya que recopilan datos sobre el sistema y las aplicaciones. Sin embargo, dentro de los node exporters, los **collectors** desempeñan un papel fundamental al recopilar información específica, como el uso de la CPU o la memoria, y proporcionar a los node exporters.

Los usuarios tienen la flexibilidad de activar o desactivar estos **collectors** según sus necesidades y preferencias. Por ejemplo, si un usuario está interesado únicamente en monitorear el rendimiento de la CPU, puede activar el **collector** correspondiente y desactivar los demás.

En resumen, los collectors son componentes clave del node exporter que permiten la **recopilación de datos de métricas específicas** para su análisis posterior en Prometheus.

La integración de Prometheus con Grafana proporciona una solución completa para la **monitorización** y **observabilidad** de sistemas, permitiendo a los usuarios visualizar métricas en tiempo real, crear paneles personalizados y configurar alertas para mantener la estabilidad y el rendimiento del sistema.







2.4. Gitlab + Gitlab Runner como CI/CD

GitLab, al igual que GitHub, es un gestor de repositorios Git que facilita la colaboración en proyectos de desarrollo de software. Escrito en Ruby y Go, esta plataforma de código abierto es gratuita para uso personal y permite que equipos trabajen juntos en un mismo proyecto, proponiendo cambios y revirtiéndolos si es necesario.

Desde su versión 10.0, GitLab ha evolucionado más allá de ser solo un repositorio Git, convirtiéndose en una plataforma que abarca todo el ciclo de vida del desarrollo de software, ofreciendo una visión completa de DevOps.

Esta integración abarca desde la planificación y gestión del código fuente hasta el monitoreo y la seguridad, proporcionando una experiencia unificada para los usuarios.



Una función nativa de GitLab es la integración continua y la entrega continua (CI/CD), que permite el desarrollo, las pruebas y el despliegue continuos de aplicaciones.

Esta característica es impulsada por GitLab CI/CD, una herramienta que ejecuta trabajos de manera continua en una canalización de desarrollo.





El componente clave de GitLab CI/CD es el **GitLab Runner**, una aplicación que ejecuta trabajos definidos en la canalización de CI/CD.

El GitLab Runner puede ejecutarse en diferentes entornos, como máquinas virtuales, contenedores **Docker** o incluso dispositivos físicos, y se encarga de llevar a cabo las tareas de construcción, prueba y despliegue definidas en la configuración de CI/CD.

El proceso que sigue este disparador o **trigger** es el siguiente. Primero, el administrador o desarrollador en un entorno de desarrollo realiza un **commit** en el repositorio correspondiente. Tras ello, Gitlab avisa a este GitLab Runner y lee un fichero que contiene las fases de la implantación de la aplicación, incluyendo instalación de las dependencias, los **test jobs**...



Una vez termine dicho proceso, podrá dar **success** cuando se haya completado correctamente y ya tendríamos en correcto funcionamiento la aplicación en un almacenador de imágenes como **Docker**, **Podman**... y **error** si nos falta algo en nuestro código o cualquier error que deberemos consultar en sus respectivos logs.

Por ello, **GitLab** proporciona una plataforma completa para la gestión de proyectos de desarrollo de software, desde la planificación hasta la entrega, con una amplia gama de herramientas integradas como **GitLab CI/CD** y el **GitLab Runne**r, que permiten la automatización y la mejora continua del proceso de desarrollo.







3. Escenario que se ha realizado

Antes de presentar el escenario, es necesario explicar los tipos de estructuras que se encuentran en Rancher. Una vez comprendidos, procederemos a analizar el esquema que he desarrollado.

3.1. Tipos de estructuras en Rancher

Antes de explorar los distintos tipos de arquitecturas en Rancher, es importante entender una recomendación clave de la plataforma: **al instalar Rancher en un solo nodo, este debe estar separado de los clústeres descendientes**. Ésto garantiza un despliegue sin conflictos y optimizado del clúster.

Los **clústeres de usuario** en Rancher representan entornos descendientes de Kubernetes donde se ejecutan aplicaciones y servicios específicos del usuario. Esta separación de clústeres habilita que podamos tener una gestión eficiente y permite una distribución adecuada de las cargas de trabajo.

Además, si se utiliza una instalación de **Docker** en Rancher, es un factor importante que el nodo servidor de Rancher esté también apartado de los clusters descendientes. Esta separación contribuye a la **estabilidad** y **eficiencia** del entorno Docker en Rancher. En definitiva, seguir estas pautas de separación entre el nodo servidor, los clusters descendientes y los clusters de usuario es clave para una gestión fluida y sin conflictos en el entorno de Rancher.

En Rancher, pueden coexistir 2 tipos de arquitecturas de instalación que son las siguientes:

- Separación del servidor Rancher de nodo único y los clústeres de usuarios: el cual se compone de los usuarios de K8s que se manejan de manera escalable desde el servidor de Rancher. A este servidor se accede desde la misma red local directamente.





Separación de servidores Rancher de alta disponibilidad y clústeres de usuarios: éste es similar al anterior, la principal diferencia es la implementación de un balanceador de carga para controlar las peticiones a este mismo servidor.



Normalmente, **Rancher nos recomienda instalar un balanceador de carga** para nuestro servidor para que goce de la alta disponibilidad de su clúster de K8s. Ésto es prioritariamente por los datos de este clúster orquestado por Rancher.

Por ello, **una instalación con alta disponibilidad** nos proporciona que el balanceador sirve como único punto de contacto para los clientes, distribuyendo el tráfico de red entre múltiples servidores en el clúster y ayudando a evitar que cualquier servidor se convierta en un punto de falla del mismo.





Rancher no recomiendo instalar Rancher en un solo contenedor Docker porque si el nodo deja de funcionar, no habrá copia de los datos del clúster disponibles en otros nodos pero... existe una **excepción**.

A la hora de crear el contenedor Docker, le añadimos un **volumen** el cual guarda simultáneamente todo el contenido sobre nuestra aplicación Rancher garantizando persistencia y seguridad en los datos eliminando por completo la idea de contenedor con **almacenamiento efímero**. Con esto dicho, ya sabemos con antelación cuál va a ser la intención de mi proyecto integrado.

Ya visto los 2 tipos de arquitecturas de instalación en Rancher, pasamos a ver los **2 tipos de instalación de clusters de K8s** que se pueden montar. Estos son los siguientes:

 Instalación de un clúster de RKE K8s: En una instalación de RKE, los datos del clúster se replican en cada uno de los tres nodos etcd del clúster, lo que proporciona redundancia y duplicación de datos en caso de que uno de los nodos falle. Una imagen de cómo sería un clúster RKE Kubernetes que ejecuta Rancher Management Server:



Nota: **Rancher Kubernetes Engine** (RKE) es una distribución de Kubernetes certificada por CNCF que se ejecuta completamente dentro de contenedores Docker.





 Instalación de un clúster de K3s K8s: Una opción para el clúster de Kubernetes subyacente es utilizar un clúster montado previamente o no (ya que lo podemos crear desde interfaz gráfica de Rancher) de K3s Kubernetes. Con los conocimientos adquiridos sobre K3s anteriormente, pasamos a ver la arquitectura de un clúster de Kubernetes K3s que ejecuta Rancher Management Server:



Cada uno de estos clusters, mantienen unas características en cuanto a funciones y roles distintas entre ellos.

En los clusters de RKE se deben tener tres nodos y cada nodo debe tener las tres funciones de Kubernetes: **etcd**, **plano de control** (controlplane) y trabajador (worker).

En los clusters de K3s hay dos tipos de nodos: **nodos de servidor y nodos de agente**. Tanto los servidores como los agentes pueden tener cargas de trabajo programadas en ellos. Los nodos del servidor ejecutan el maestro de K8s.

Conociendo las diferencias de estructuras de instalación y de clusters, voy a presentar en que se basará mi proyecto.





3.2. Escenario elegido con todas sus características

Con respecto al escenario que quiero desplegar, me basaré en una estructura con las. Por ello, se montará un cluster de **K3s** ya que nos proporciona un alto rendimiento consumiendo una cantidad de recursos mínima.

Se necesitará un nodo manager que tendrá instalado Rancher y nos controlará a nuestro nodo master de **K3s** que tendrá instalada las demás aplicaciones. Estos nodos se instalarán en KVM.

Seguido de esto, **Rancher** (implementado en Docker) con un balanceador de carga (Nginx) para poder hacer peticiones a la aplicación desplegada mediante **GitLab** (desplegado en Rancher) con su respectivo **Runner** y controlado mediante las métricas de **Grafana** y **Prometheus** (desplegado en Rancher).

Con estos componentes, el esquema se verá de la siguiente manera:



Con dichas características, podemos **ver** la **funcionalidad y el desempeño de cada una de estas herramientas de producción** como la integración continua mediante GitLab y la observabilidad y monitorización mediante Grafana y Prometheus **en un clúster local con alta disponibilidad** con K3s. Tras esto, pasamos a la instalación.





3.3. Instalaciones

En los siguientes apartados, realizaremos las instalaciones de todos los servicios y sistemas que utilizaremos en el proyecto.

3.3.1. Nodos mediante KVM

Para poder montar nuestro clúster de alta disponibilidad en Rancher con K3s, nos hace falta tener nuestros nodos montados correctamente.

Destacar que las máquinas de este esquema como la máquina host donde hemos instalado el gestor de máquinas virtuales, serán máquinas **Ubuntu 22.04 Jammy Jellyfish** pero estos comandos también son válidos para distribuciones como **Debian**.

Para desplegar el esquema, actualizaremos la máquina para poder tener los repositorios actualizados. Tras esto, ejecutaremos el 2 comando para ver que nuestro sistema cumpla con los requisitos de KVM y es necesario para evitar errores que puedan resultar en una instalación incompleta.

\$ sudo apt update -y && sudo apt upgrade \$ sudo egrep -c '(vmx|svm)' /proc/cpuinfo

Cuando ya lo tengamos instalado, pasamos a la creación del escenario. Para poder desplegar nuestro esquema, deberemos crear una máquina en nuestro gestor de máquinas virtuales con la versión de **Ubuntu** correspondiente.

Si no tenemos instalado **KVM**, podemos instalarlo de la siguiente manera:

```
$ sudo apt-get install qemu-kvm libvirt-daemon-system libvirt-clients
bridge-utils virt-manager
$ sudo usermod -aG libvirt $USERNAME
$ sudo usermod -aG kvm $USERNAME
```





Además de instalarlo, tenemos que darle permisos para que el usuario sin privilegios pueda tener acceso a cualquier máquina virtual, volumen lógico... etc

Con esto comentado, pasamos a la instalación de una máquina Ubuntu **22.04** y cuando la tengamos la clonamos 3 veces para poder realizar el proyecto.

Subrayar que la máquina principal, tendrá un almacenamiento de 25GB y un uso de memoria máximo de 4GB de RAM.



Algunos retogues pueden ser deshabilitar el uso de contraseña para el acceso de permisos de root, cambiar el nombre del hostname en la terminal para reconocer las máquinas y actualizar los repositorios para poder salir a internet ya que la versión de Ubuntu 22.04 está desactualizada (actualmente por la 24.04 Noble Numbat).



Desplegamos las máquinas y accedemos a ellas para ver que se han creado correctamente. Con las máquinas creadas y operativas, pasamos al despliegue de Rancher y el clúster de K3s.





3.3.2. Rancher y cluster local de K3s

En primer lugar, instalamos Rancher en la máquina **manager**, en la máquina donde desplegamos **Gitlab** y en la máquina de Gitlab para poder hacer el login al **Registry privado** con mediante **Kaniko**.Acto seguido, conectamos el servidor **master** mediante un nodo de K3s.

Para instalar Rancher, utilizaremos Docker como herramienta de despliegue de la aplicación. La instalación de Docker se habilitará añadiendo unos repositorios que instalan **docker-ce**. Esto nos servirá para poder hacer uso de los comandos de Docker.

```
$ sudo apt-get update
$ sudo apt install 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
$ sudo apt-get update
$ sudo apt install docker-ce
$ sudo usermod -aG docker ${USER}
```

A continuación, tendremos que utilizar un dominio propio de este modo podremos tener una clave certificada por una **CA**. Si quieres crear una propia y firmar con una Entidad Certificadora, también se puede realizar.

Para poder copiar las claves de mi host a mi máquina manager, las pasamos mediante **scp** desde mi host a esta misma. Un ejemplo:

\$ scp /etc/ssl/rancher/* pepe@[ip_ext_máquina_manager]:

Seguidamente, creamos en la máquina manager el siguiente directorio y copiamos los archivos mediante **scp** a la ruta indicada:



```
<u>A</u>
```

```
$ sudo mkdir -p /etc/ssl/rancher
$ sudo cp [nombredominio]_ssl_certificate.cer /etc/ssl/rancher/
$ sudo cp _.[nombredominio]_private_key.key /etc/ssl/rancher/
```

Con los certificados ubicados en mi máquina manager, pasamos a la instalación de Rancher. Así podremos unificar las claves que tenemos en nuestra máquina manager y el contenedor **Docker**, tenemos que crear un volumen con el comando de **Docker Engine** para que se guarde automáticamente. Para ello, utilizaremos el siguiente comando:

```
$ sudo docker run -d --name rancher --restart=unless-stopped \
    -p 8080:80 -p 8443:443 \
    -v
/etc/ssl/rancher/pepepfoter15.es_ssl_certificate.cer:/etc/rancher/ssl/pepepf
oter15.es_ssl_certificate.cer \
    -v
/etc/ssl/rancher/_.pepepfoter15.es_private_key.key:/etc/rancher/ssl/_.pepepf
oter15.es_private_key.key \
    -v /var/lib/docker/rancher:/var/lib/rancher \
    --privileged \
    rancher/rancher:stable
```

Automáticamente Docker, extrae la imagen de Rancher ubicada en **Docker Hub**, la descarga y crea el contenedor con la misma.

```
run -d --name rancher --restart=unless-stopped \
     8080:80 -p 8443:443 \
   v /etc/ssl/rancher/pepepfoter15.es_ssl_certificate.cer:/etc/rancher/ssl/pepepfoter15.es_ssl_certif
icate.cer \
  -v /etc/ssl/rancher/_.pepepfoter15.es_private_key.key:/etc/rancher/ssl/_.pepepfoter15.es_private_ke
  .v /var/lib/docker/rancher:/var/lib/rancher \
    --privileged \
    rancher/rancher:stable
unable to find image 'rancher/rancher:stable' locally
stable: Pulling from rancher/rancher
53198d61f590: Pull complete
21251ce9e27c: Pull complete
11eb5e81edd1: Pull complete
 5e3489ac5ae: Pull complete
1563d5833068: Pull complete
c4c5e72c3608: Pull complete
664c371e3589: Pull complete
37a2a72f797f: Pull complete
44e69243627d: Pull complete
9d4fe59c24b9: Pull complete
38b50947c238: Pull complete
676e3bff064c: Pull complete
1c038cbe2800: Pull complete
9f2fae27c7fc: Pull complete
fea0e93002f2: Pull complete
9ca5e34b294c: Pull complete
4cb5d07e072e: Pull complete
4ff9191461b1: Pull complete
616e7747783a: Pull complete
c672d755205c: Pull complete
Digest: sha256:6e3ea2fdc11c98de7fa71ef2dc434396c9e0255f8fbbc56de1f53752a5174458
Status: Downloaded newer image for rancher/rancher:stable
f2dc95660bd66123<u>4</u>702badeddbded831bbb2774ce2e94363cb05ed8ca798047
     @manager:~$
```





Tras éllo, ya tendríamos instalado Rancher como contenedor. Para verificar que la máquina está funcionando correctamente, abrimos el navegador e ingresamos esta URL con la IP externa que utilizamos para pasar los certificados a nuestra máquina manager:

https://[ip_ext_máquina_manager]:8443/

Cuando accedemos, aparecerá un mensaje indicando que no se reconoce el certificado. Para continuar, seleccionamos Avanzado y luego Aceptar el riesgo y continuar.



Seguido de este proceso, aparecerá la interfaz de Rancher, la cual nos pedirá que ejecutemos algunos comandos para poder acceder mediante la contraseña de Bootstrap.

Para comenzar, nos pide que veamos el **ID** de nuestro contenedor y, seguido de esto, consultemos la contraseña con los siguientes comandos:





\$ sudo docker ps \$ sudo docker logs [id-	contenedor] 2>&	1 grep "Boo	tstrap Passwo	ord:"
Todo ésto en nuestra	máquina manage	er.		
pepe@manager:~\$ sudo docker ps				
CONTAINER ID IMAGE	COMMAND	CREATED N	STATUS AMES	PORTS
f2dc95660bd6 rancher/rancher:s ->80/tcp, :::8080->80/tcp, 0.0.0 pepe@manager:~\$ sudo docker logs 2024/06/11 11:49:26 [INFO] Boots pepe@manager:~\$	table "entrypoint.s).0:8443->443/tcp, ::: f2dc95660bd6 2>&1 trap Password: tkj2nl	h" 6 minutes ag 8443->443/tcp r grep "Bootstrap lvvgqqgvkmfmb2xrk	o Up 6 minutes ancher Password:" f8vgmks8wh5tnggdp	0.0.0.0:8080 jv8jqm8z6vfb5m

Como podemos observar, tenemos que copiar la clave y pegarla en la página desplegada en nuestro navegador.

Howdy! Welcome to Rancher							
It looks like this is your first time visiting Rancher; if you pre-set your own bootstrap password, enter it here. Otherwise a random one has been generated for you. To find it:							
For a "docker run" installation:							
• Find your container ID with docker ps, then run:							
docker logs <u>container-id</u> 2>&1 grep • "Bootstrap Password:"							
For a Helm installation, run: kubectl get secretnamespace cattle-system bootstrap-secret -o go-							
base64decode}}{{"\n"}}'							
Password Show							
Log in with Local User							

Seguido de ésto, podemos cambiar la contraseña del usuario administrador si queremos o la podemos dejar por defecto y podemos cambiar la URL a la cual debería usar Rancher en la instalación.

Un dato muy importante, es que todos los nodos deben de ser accesibles a esta URL, por ello, nos conviene no modificarlo.



Por último, aceptamos los términos y condiciones de esta aplicación para el usuario y continuamos.

g password for the default $\overline{\mathtt{admin}}$ user. We suggest using this random one generated just
for you, but enter your own if you like.
Сору
installation? All the nodes in your clusters will need to be able to reach this.
s to help us improve rancher
I User License Agreement & Terms & Conditions
s to help us improve rancher I User License Agreement & Terms & Conditions

Y con ésto, ya tendríamos Rancher instalado.

$\leftarrow \ \ \rightarrow$	С	🔿 🔒 https://192.1	68.122.3 :8443/dashboa	rd/home			
	FRANCHEI	ť					
f							
'				Welcome to Ra	ncher		
						Ŷ	Ģ
	Learn m						
	You can change what you see when you login via preferences						Preferences X
	Clusters				Manage	rt Existing Create	
	State ≎	Name 🗘	Provider 🛇	Kubernetes Version	СР∪≎	Memory 🗘	Pods 🗘
			Local K3s	v1.28.6+k3s2	4 cores	3.82 GiB	6/110





Al tener Rancher instalado, ya tenemos 1 paso, nos falta la instalación del nodo de K3s. Para ello, instalamos en la máquina **master** con el siguiente comando:

\$ sudo apt-get update \$ curl -sfL https://get.k3s.io sh -						
<pre>pepe@master:~\$ curl -sfL https://get.k3s.io sh - [INF0] Finding release for channel stable [INF0] Using v1.29.5+k3s1 as release</pre>						
<pre>[INFO] Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.29.5+k3s1/sha256sum-amd64 .txt [INFO] Downloading binary <u>https://github.com/k3s-io/k3s/releases/download/v1.29.5+k3s1/k3s</u> [INFO] Verifying binary download [INFO] Installing k3s to /usr/local/bin/k3s</pre>						
<pre>[INF0] Skipping installation of SELinux RPM [INF0] Creating /usr/local/bin/kubectl symlink to k3s [INF0] Creating /usr/local/bin/crictl symlink to k3s [INF0] Creating /usr/local/bin/ctr symlink to k3s [INF0] Creating killall script /usr/local/bin/k3s-killall.sh</pre>						
<pre>[INFO] Creating uninstall script /usr/local/bin/k3s-uninstall.sh [INFO] env: Creating environment file /etc/systemd/system/k3s.service.env [INFO] systemd: Creating service file /etc/systemd/system/k3s.service [INFO] systemd: Enabling k3s unit</pre>						
vice. [INFO] systemd: Starting k3s pepe@master:~\$						

Ya instalado, pasamos a importar el clúster desde la interfaz gráfica, indicando Import Existing.

Learn more	about the impro	ovements and new cap	pabilities in this version.			
You can cha	ange what you se	e when you login via	preferences			Preferences
Clusters			Manage	Import Existing	Create Filter	
Clusters State 🗘	1 Name¢	Provider 🗘	Manage	Import Existing	Create Filter	Pods 🗘

Nos pedirá el tipo de cluster a importar y como podemos observar, están los clústers EKS, AKS y GKE que comentamos en el inicio de la memoria.



A nosotros nos interesa presionar Generic para importar un clúster de K3s y indicamos el nombre y descripción de éste:

Clusters		
Cloud Credentials		Cluster: Import
Drivers		Posistar an aviating dustar in a bastad Kubarnatas providar
RKE1 Configuration	>	
Advanced	>	Import any Kubernetes cluster
		Generic

Cluster Management						
Clusters						
Cloud Credentials		Cluster: Import Gener				
Drivers						
RKE1 Configuration	>	Import Harvester Clusters via Virtu				
Advanced	>					
		Cluster Name * cluster-k3s		Cluster Description Cluster para el PI		
		Member Roles	Memt	per Roles		
		U: Labels & Annotations	User		Role	
				Default Admin (admin) .ocal	Cluster Owner	
			[Add	l	

Cuando ya tengamos los nombres, hacemos clic en **Crear** para establecer la base que nos permitirá importar nuestro clúster.

Luego, Rancher nos pedirá que indiquemos el tipo de certificado que tiene. En nuestro caso, debemos elegir la **segunda opción**, ya que nuestro certificado no es confiable porque está en la máquina manager.



Cluster: cluster-k3s Pending Namespace: fileet-default Age: 1 secs	Explore	Detail	Config	:
This resource is currently in a transitioning state, but there isn't a detailed message available.				
Provisioner: Imported				
Provisioning Log Registration Conditions Recent Events Related Resources				
You should not import a cluster which has already been connected to another instance of Rancher as it will lead to data corruption.				
Run the kubectl command below on an existing Kubernetes cluster running a supported Kubernetes version to import it into Rancher:				
kubectl apply -f https://192.168.122.3:8443/v3/import/57khmg7bm2ctb4kd4xtqw1jzfbt1k8kgv69wjdzd82pn9rqwj5czxw_c-m-4hqm267j.yaml				
If you get a "certificate signed by unknown authority" error, your Rancher installation has a self-signed or untrusted SSL certificate. Run the command below instead to	oypass the cer	tificate verifi	cation:	
curlinsecure -sfL https://192.168.122.3:8443/v3/import/57khmg7bm2ctb4kd4xtqw1jzfbt1k8kgv69wjdzd82pn9rqwj5czxw_c-m-4hqm267j.yaml ku	bectl apply			
If you get permission errors creating some of the resources, your user may not have the cluster-admin role. Use this command to apply it:				
kubectl create clusterrolebinding cluster-admin-bindingclusterrole cluster-adminuser <your from="" kubeconfig="" username="" your=""></your>				

Copiamos este comando y lo ejecutamos en nuestra máquina máster como root. Veremos que se aplican todos los service, deployments, secret, namespaces...

\$ sudo su \$ curlinsecure -sfL https://[ip_ext_máquina_manager]:8443/[url].yaml kubectl apply -f -
pepe@master://ome/pepe#
<pre>root@master:/home/pepe# curlinsecure -sfL https://192.168.122.3:8443/v3/import/57khmg7bm2ctb4kd4x qwljzfbtlk8kgv69wjdzd82pn9rqwj5czxw_c-m-4hqm267j.yaml kubectl apply -f - clusterrole.rbac.authorization.k8s.io/proxy-clusterrole-kubeapiserver created clusterrolebinding.rbac.authorization.k8s.io/proxy-role-binding-kubernetes-master created namespace/cattle-system created serviceaccount/cattle created clusterrolebinding.rbac.authorization.k8s.io/cattle-admin-binding created secret/cattle-credentials-c1db470 created clusterrole.rbac.authorization.k8s.io/cattle-admin created Warning: spec.template.spec.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nod SelectorTerms[0].matchExpressions[0].key: beta.kubernetes.io/os is deprecated since v1.14; use "kube netes.io/os" instead deployment.apps/cattle-cluster-agent created service/cattle-cluster-agent created</pre>

Por último, deberemos esperar unos segundos y en la misma interfaz que lo dejamos, refrescamos en el navegador y vemos que se ha conectado correctamente.



		4	Å	Q
Explore	Detail	Config	:	

Cluster: cluster-k3s (Active) Namespace: fleet-default Age: 1 secs			Explore	Detail Config :
Provisioner: K3s				
Machine Pools Provisioning Log Conditions Recent Event				
□ State ≎ Name ♀ Node	External/Internal IP 🛇	os≎	Roles 🗘	Age 🗘
Not in a Pool				
Active machine-86rq2 maste	-/192.168.122.31	Linux	Control Plane	33 secs

Con ésto, ya tendríamos creado e importado el cluster de alta disponibilidad con K3s en Rancher.

<u> </u> cluster-k3	;							Only User Namespa	ces `	´ 1	<u>ک</u>	o 1	: S
Cluster	~												
Projects/Namesp	aces	Cluster Da	ashboard										
Nodes		Provider: K3s	Kubernetes Version: v1.29	.5 +k3s1 Created:	3.9 mins ag	o					Install Mor	itoring 📕 Ad	ld Cluster Badge
Cluster and Proje	ct Members												
Workloads	(iiii)												
Apps		298	Total Reso	urces		1	Node		7	De	ployments		
Service Discovery]
Storage		Capacity											
Policy													
More Resources		Pods				CPU			Memory				
		Used 8/11	0	7	.27%	Reserved 0.2/4 cores		5.00%	Reserved 0.14	/ 3.82 GiB			3.66%
									Used 073826				0.00%
]
		✓ Etcd	✓ Scheduler	✓ Controller Ma	inager								
													ull events list
		Reason 🗘	Object 🗘	Me	ssage 🗘			Name 🗘			Date 🗘		
		Pulled		bdtp Cor	ntainer ima	ge "rancher/shell:v0.1.24" alre	ady present on machine				Tue, Jun	11 2024 2:11	:26 pm
		Created		:bdtp Cre	ated contai	ner helm					Tue, Jun	11 2024 2:11	:26 pm
		Started		:bdtp Sta	rted contaiı	her helm					Tue, Jun	11 2024 2:11	:26 pm
& Cluste	r Tools	Pulled		bdtp Cor	ntainer imag	ge "rancher/shell:v0.1.24" alre	ady present on machine				Tue, Jun	11 2024 2:11	:26 pm
v284		Created		bdtp Cre	ated contai	ner proxy					Tue, Jun	11 2024 2:11	:26 pm
		Started	Pod belm-operation-k	ibdto Sta	rted contair	Der Drovy		helmionerati	on-khdtn 17d7f2051	7432669	Tue lun	11 2024 2-11	•26 nm
THE RANCE	IER [®]												:
						Welcome to I	Rancher			ł			•
											4		
										_			•
Lear	n more about the	improvements and ne	ew capabilities in this version.										s new in 2.8
You	can change what y	vou see when you logi	in via preferences						X Link				
Clust	ers 2					Manage	nport Existing Create						
State	i Name 🤇		Provider 🗘	Kubernetes Version		ср∪≎	Memory 🗘	Pods					
(Act			Imported	v1.29.5+k3s1		4 cores	3.82 GiB	8/110					
			Local										
			K3s	v1.28.6+k3s2		4 cores	3.82 GiB	6/110					





3.3.3. Grafana + Prometheus

Para la correcta instalación de Grafana y Prometheus, tendremos que instalar el chart de ambas herramientas mediante Helm. Para este proceso, tendremos que utilizar la terminal con la herramienta Helm. La instalación se hará en la máquina Master, por ello, instalaremos dicha herramienta ejecutando el script que descargamos.

\$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 \$ sudo chmod 700 get_helm.sh \$ sudo ./get_helm.sh

Seguido de esto, nos descargamos el repositorio donde vamos a sacar los **charts** de Grafana y Prometheus para poder instalarlo. Para ello, utilizaremos la herramienta Helm (como superusuario):

\$ helm repo add grafana https://grafana.github.io/helm-charts \$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts \$ helm repo update

La instalación de las aplicaciones es bastante sencilla ya que disponemos de un solo comando para cada servicio para instalarlo que haciendo referencia a nuestro **kubeconfig** y al mismo repositorio, conseguimos instalarlo correctamente.

<pre>\$ helm install prometheus prometheus-community/prometheuskubeconfig</pre>								
/etc/rancher/k3s/k3s.yaml								
\$ helm install grafana grafana/grafanakubeconfig								
/etc/rancher/k3s/k3s.yaml								

Tras ésto, para las pruebas referentes a nuestras máquinas, crearemos 2 services para ver ambas aplicaciones mediante NodePort.

\$ kubect1 expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext \$ kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-ext --name=grafana-node-port-service





Para poder observar las aplicaciones de manera correcta, tendremos que acceder a nuestra aplicación en Rancher en el apartado de Services y copiar el puerto de las aplicaciones. Para poder ver las aplicaciones de manera correcta, ponemos la ip del nodo y el puerto copiado anteriormente:

Services 🕸				Create
Download YAML Delete 1 selected		l	E Filt	
State 🗘 Name 🗘	Target 🗘	Selector 🗘	Туре 🗘	Age 🗘
Namespace: default				
C (Active) grafana	service 7 [±] 3000/TCP	app.kubernetes.io/instance≃grafana app.kubernetes.io/name≈grafana	Cluster IP	2.4 hours
Active grafana-node-port-service	[Any Node]:31789	app.kubernetes.io/instance=grafana app.kubernetes.io/name≃grafana	Node Port	2.4 hours :
Active kubernetes	https T^L 6443/TCP		Cluster IP	2.7 hours :
CActive prometheus-alertmanager	10.43.236.69:5093 <mark>-¹ http/TCP</mark>	app.kubernetes.io/instance=prometheus app.kubernetes.io/name=alertmanager	Cluster IP	2.4 hours :

Y vemos que las aplicaciones están perfectamente instaladas 🙂.

🚏 Rancher	🧔 Grafana 🛛 🗙 🗙		
$\leftarrow \ \rightarrow \ \mathbf{G}$	192.168.122.31:31789		
		Welcome to Grafana	
		Email or username	
		email or username	
		Password	
		Log in	
		Forgot your password?	

- Grafana:





Para poder acceder a la aplicación, tenemos que obtener la contraseña de administrador y lo conseguiremos mediante este comando:

\$ kubectl get secret --namespace default grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo

Con éllo, copiamos la contraseña y con el usuario admin accederemos a la aplicación:



Prometheus:



Para la configuración y conexión de estas herramientas, se hará en apartado 3.4.2.





3.3.4. Gitlab y Gitlab Runner

Para la correcta integración de Gitlab en nuestro clúster de K3s, tendremos que instalar la aplicación en nuestra máquina Gitlab dedicada. Para este proceso, primero instalamos la aplicación y posteriormente el Runner.

Para ello, descargamos del repositorio de Gitlab oficial el script de instalación para obtener este paquete gitlab-ce (aparte de instalar otros paquetes como curl, postfix para el correo... etc).

\$ sudo apt-get update \$ sudo apt-get install -y curl openssh-server ca-certificates tzdata perl \$ sudo apt-get install -y postfix \$ curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb .sh | sudo bash

Cuando lo tengamos, pasamos a la instalación de la aplicación con este comando, especificando la IP externa de la máquina de gitlab ya que vamos a utilizar el protocolo **HTTP** sin nombre de dominio ninguno.

\$ sudo EXTERNAL_URL="http://[ip_externa_máquina_gitlab]" apt-get install
gitlab-ce

Este proceso se puede demorar unos minutos ya que tiene que montar muchos servicios por detrás para poder desplegarlo.







Cuando este finalice, accederemos a la URL indicada en la instalación y comprobaremos que se ha instalado correctamente.

🚏 Rancher	×	🌀 Grafana	×	🎒 Prometheus Time Series 📯 🦊	Sign in - GitLab	×	+
$\leftarrow \rightarrow \mathbf{G}$		0 🍐 192.168.122.208/0	ısers	/sign_in			
							V
					GitLab (Com	nmunity Edition
				U	Jsername or primary email		
				P	Password		
							۲
				_	_		Forgot your password?
] Remember me		
						S	Sign in
					Don't have ar	n acc	ount yet? Register now

Si queremos conocer la contraseña del usuario root con el acceso a la aplicación, tendremos que ver el contenido de este fichero:

\$ sudo cat /etc/gitlab/initial_root_password

Cuando tengamos esta contraseña, con el usuario root accederemos a la aplicación de manera correcta y ordenada.

🚏 Rancher		🧔 Grafana	× . (Prometheus Time S	eries 🗵 🦊 Projects - GitLab					🙁 Nave
$\leftarrow \ \ \rightarrow \ \ \mathbf{G}$		🔿 훱 어 192.168.122.20	98					\$a €	\$
₩ □	+ C	Your work / Projects							
D 11	Ø								
Q Search or go	to				Weld	come to GitLab			
Your work					Faster relea	ases. Better code. Less pain.			
Projects									
% Groups									
D Issues									
ំំង Merge requests	3	•	E	Create a project	re your code, access	A	Create a group	and	
🖂 To-Do List				issues, wiki and other featu	res of GitLab.		people.	anu	
Milestones									
🐰 Snippets									
S Activity									
							Configure Gitl ab		
				Add people Add your team members ar	ıd others to GitLab.		Make adjustments to how your GitLab instan set up.	ice is	

Con esto, ya tenemos desplegado GitLab.





Pasamos a la instalación de nuestro **Gitlab Runner**. Este runner estará instalado en la misma máquina de la aplicación para hacer la conexión a este mismo de manera rápida y sencilla.

Por esto mismo, lo primero que deberemos realizar es descargar del repositorio de Gitlab oficial el script de instalación para el paquete **gitlab-runner**. Tras esto, instalamos el runner con nuestro gestor de paquetería correspondiente.

\$ curl -L
"https://packages.gitlab.com/install/repositories/runner/gitlab-runner/scrip
t.deb.sh" | sudo bash
\$ sudo apt-get install gitlab-runner

pepe@gitlab:~\$ curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/scrip deb.sh" | sudo bash % Total % Received % Xferd Average Speed Time Time Time eed Time Time Time Current oad Total Spent Left Speed 0--:--:- 21857 Dload Upload 0 21825 0 -Detected operating system as Ubuntu/jammy. Checking for curl... Detected curl... Checking for gpg... Detected gpg... Running apt-get update... done. Installing apt-transport-https... done. Installing /etc/apt/sources.list.d/runner_gitlab-runner.list...done. Importing packagecloud gpg key... done. Running apt-get update... done. The repository is setup! You can now install packages. pepe@gitlab:~\$ sudo apt-get install gitlab-runner Leyendo lista de paquetes... Hecho Creando árbol de dependencias... Hecho eyendo la información de estado... Hecho Paquetes sugeridos: docker-engine Se instalarán los siguientes paquetes NUEVOS: gitlab-runner 0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 192 no actualizados. Se necesita descargar 493 MB de archivós. Se utilizarán 539 MB de espacio de disco adicional después de esta operación. Des:1 https://packages.gitlab.com/runner/gitlab-runner/ubuntu jammy/main amd64 gitlab-runner amd64 Des:1 https://packages.gitlab.com/runner/gitlab-runner/ubuntu jammy/main amd64 gitlab .0.0-1 [493 MB] Descargados 493 MB en 15s (32,3 MB/s) Seleccionando el paquete gitlab-runner previamente no seleccionado. (Leyendo la base de datos ... 262928 ficheros o directorios instalados actualmente.) Preparando para desempaquetar .../gitlab-runner_17.0.0-1_amd64.deb ... Desempaquetando gitlab-runner (17.0.0-1) ... Configurando gitlab-runner (17.0.0-1) ... Gitlab Runner: creating gitlab-runner... Home directory skeleton not used Runtime platform arch=amd64 os=linux pid=20080 rev Runtime platform ersion=17.0.0 arch=amd64 os=linux pid=20080 revision=44feccdf gitlab-runner: the service is not installed arch=amd64 os=linux pid=20089 revision=44feccdf untime platform gitlab-ci-multi-runner: the service is not installed Runtime platform ersion=17.0.0 Runtime platform ersion=17.0.0 arch=amd64 os=linux pid=20114 revision=44feccdf arch=amd64 os=linux pid=20177 revision=44feccdf Check and remove all unused containers (both dangling and unreferenced) including volumes. Total reclaimed space: OB pepe@gitlab:~\$





Este proceso de instalación puede tardar algunos minutos. Cuando se termine la instalación tendremos que crear el Runner mediante un **token de verificación**, pero primero, tendremos que crear un nuevo proyecto.

Primero, seleccionamos Create a project -> Create blank project



Seguido de esto, ponemos un nombre al proyecto y seleccionamos un **namespace** para el proyecto. Con esto comentado, creamos el proyecto.

$\leftarrow \rightarrow G$ C	2 192.168.122.208/projects/new#blank_project 🗴 🗘 👳						
⊌ □ + ○	Your work / Projects / New project / Create blank project						
C Search or go to	Create blank project Create a blank project to store your files, plan your work, and collaborate on code, among other things.						
Projects	Project name						
% Groups	mi-primera-app						
D Issues	Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.						
% Merge requests >	Project URL Project slug						
🗹 To-Do List	http://192.168.122.208/ root ~ / mi-primera-app						
Milestones	Visibility Level (?)						
🐰 Snippets	O ⊕ Private						
S Activity	Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group. $\bigcirc \Phi$ internal						
	The project can be accessed by any logged in user except external users.						
	C the project can be accessed without any authentication.						
	Project Configuration						
	Initialize repository with a README Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.						
	Enable Static Application Security Testing (SAST) Analyze your source code for known security vulnerabilities. Learn more.						
	Create project Cancel						

Cuando lo creemos, ya tendremos nuestro proyecto en blanco a la espera de poner nuestro contenido pero... ahora mismo esto no nos interesa.



🐨 Rancher	× 🌀	Grafana	× 🕚 Prometheus Time Series 🗴 🦊 Adm	ninistrator / mi-prime × +	
$\leftarrow \ \rightarrow \ \mathbf{G}$	0	192.168.122.208/ro	oot/mi-primera-app		
⊌ □	+ ()	Administrator / mi-prime	era-app		
D 13	$\mathbf{\nabla}$				
Q Search or go to	o		M mi-primera-app ≞		29
Project			양 main ~ mi-primera-app / + ~		History Find file Edit ~ Code ~
M mi-primera-app					
🖈 Pinned	~		Update README.md	2	44419a88 [⁶]
Issues	0			-	
Merge requests	0		Name	Last commit	Last update
සී Manage	>		M README.md	Update README.md	5 minutes ago
🛱 Plan	>				
> Code	>		README.md		
🧐 Build	>				
⊕ Secure	>		mi-primera-app		
Deploy	>		Drimor coorcomionto o CI/CI	Deen Citleh mentada en al alu	ator local con Denchor y K2a
Operate	>			o con Gillab montado en el clu	ster tocat con Rancher y K3S

Pasamos a la importación del Runner; para ello, accederemos a Settings -> CI/CD.

₩	□ +	\odot	Administrator / mi-primera-app / Repository
D	:1 E	3	main × mi-primera-app / + x
Q Sea	rch or go to		
Branches			Update README.md
Commits			Administrator authored just no
Tags			
Repositor	y graph		Name
Compare	revisions		M* README.md
Snippets			
🕼 Build		>	E README.md
① Secure		>.	
Deploy		>	mi-primera-app
Operate		>	Primer acercamiento a
🖽 Monitor		>	
山 Analyze		>	
Settings		~	
General			
Integratio	ns		
Webhook	S		
Access T	okens		
Repositor	У		
Merge re	quests		
CI/CD		×	
Packages	and registries		





Clicamos en Runners -> Expand y se nos desplegará un contenido informativo sobre qué son los runners y cómo funcionan. Con esto comentado, creamos el Runner clicando en New project runner.

🐨 Rancher 🛛 🗙 🧔	Grafana × O Prometheus Time Series × 🦊 CI/CD Settings - CI/CD - S × +	🙁 Navegación privada
$\leftarrow \rightarrow C$	원 192.168.122.208/root/mi-primera-app/-/settings/ci_cd 중 192.168.122.208/root/mi-primera-app/-/settings/ci_cd	A 12 🔍
⊌ □ + ○	Administrator / mi-primera-app / CICD Settings	
D 11 R	Customize your pipeline configuration.	
Q Search or go to	Auto DevOps	Expand
Issues	Automate building, testing, and deploying your applications based on your continuous integration and delivery configuration. How do I get started?	
Merge requests 0	Runners	Collapse
යි Manage >	Runners are processes that pick up and execute CI/CD jobs for GitLab. What is GitLab Runner?	
⊟ Plan >	Register as many runners as you want. You can register runners as separate users, on separate servers, and on your local machine.	
V> Code >	How do runners pick up jobs?	
	Runners are either:	
⊕ Secure >	(active) - Available to run jobs. (pagesd) - Not available to run jobs.	
ତ Deploy >	Tanc control which type of jobs a runner can handle. By tagging a number you make sure runners only handle the jobs they are environed to run. Learn more	
Operate >	regis control miner (pre or joos or reliner can nanae). Or augung a reinter, you make our crainers only nanate me foos why are experient and in the court mater.	
H Monitor >	Project runners Instance runners	
i Analyze →	These runners are assigned to this project. These runners are available to all groups and projects.	
Settings ~	New project runner : Enable instance runners for this project	
General		

En la creación del Runner, tendremos que indicar el nombre, marcar que este Runner pueda lanzar el **pipeline** de los jobs que no están taggeados y ponemos 600 segundos como tiempo mínimo de job.

🚏 Rancher	× 6	🗘 Grafana x 🕐 Prometheus Time Series x 🤟 Create a project runner · X + v 🏟 Navegac	ión privada
$\leftarrow \rightarrow \mathbf{G}$	0	C 👌 192.168.122.208/root/mi-primera-app/-/runners/new 🎘 😭	
₩	□ + ○	Administrator / mi-primera-app / OlCD Settings / New runner	
D n		New project rupper	
Q Search or	go to		
Project		Create a project runner to generate a command that registers the runner with all its configurations.	
M mi-primera-app			
🔊 Pinned	~	Tags	
Issues	0	Tags	
Merge requests	0	Add tags to specify jobs that the runner can run. Learn more.	_
88. Manage	,	N	
自 Plan		Separate multiple tags with a comma. For example, macos, shared.	
e Fair		Run untagged jobs Use the runner for jobs without tags in addition to tagged jobs.	
Code	,		
Ø Build	>	Configuration (optional)	
Deploy	>	Bunner description	
 Operate 			٦
Wanitar			
		Paused Stop the runner from accepting new jobs.	
⊭ Analyze	>	Protected If the the summer an alicellane for protected hereafter and.	
Settings	>	□ se une i unite on piperines la piperines in piperines	
		Use the runner for the currently assigned projects only. Only administrators can change the assigned projects.	
		Maximum job timeout	
		Maximum amount of time the runner can run before it terminates. If a project has a shorter job timeout period, the job timeout period of the instance runner is used instead.	_
		600 C	
		Enter the job timeout in seconds. Must be a minimum of 600 seconds.	
		Create runner	

Cuando tengamos esta configuración lo creamos y nos mostrará la siguiente interfaz.



😻 Rancher			Grafana	× 🌔 Prometheus Time :	Series 🗵 🤟 Register - #	1 (Ceb-ntsVw)× +				~ 🗢 M	lavegación privada 🛛 🗕
	3	0	🔒 192.168.122.208/r	oot/mi-primera-app/-/runn	ers/1/register					\$4 ☆	
₩		+ 🔘	Administrator / mi-prim	nera-app / CI/CD Settings / Re	gister runner						
D' Q Se	81 earch or go to	G 		 Runner created. 							×
Project M mi-prim Issues	iera-app	ů O		Register runn Platform Operating systems	er						
Merge i රිසි Manage	equests	•		• 👌 Linux	macOS	O Windows					
⑦ Plan √> Code Ø Build		>		Containers	🛞 Kubernetes 🗗						
① Deploy		>		GitLab Runner must be inst	talled before you can regist	ter a runner. How do I ins	stall GitLa	ab Runner?			
		>		Step 1 Copy and paste the followi	ing command into your com	nmand line to register the	ie runner.				
I Analyze	5	>		<pre>\$ gitlab-runner regi url http://192.1 token glrt-Ceb-r</pre>	ister 168.122.208 ntsVwz5jT6of6AJE						₿ <u>C</u>

Como podemos ver, tendremos que correr el Runner mediante el token generado que tendremos que copiar. El comando en cuestión para correr dicho Runner es el siguiente:

\$ sudo gitlab-runner register -n --url http://[ip_externa_máquina_gitlab] --registration-token [token_generado_gitlab] -executor docker --description "Deployment Runner" --docker-image "docker:stable" --tag-list deployment --docker-privileged

Le daremos a View Runners en nuestra aplicación (cuando ejecutemos el comando anterior), y veremos que sale el Runner en verde, es decir de manera activa.

🚏 Rancher	× 🇔	Grafana	× 🧕 Prometheus Time Series 🗵 🦊 CI/CD Setting:	• CI/CD · SC× +	V 💀 Navegaci
$\leftarrow \rightarrow \mathbf{G}$	C	🔒 192.168.122.208/rd			8 私 ☆
₩ □	+ ()	Administrator / mi-prime	era-app / CI/CD Settings		
D. 87	Ø		Project runners		Instance runners
Q Search or go to			These runners are assigned to this project.		These runners are available to all groups and projects.
issues Merge requests	0		New project runner		Enable instance runners for this project
සි Manage	>				This GitLab instance does not provide any instance runners yet. Administrators can register instance runners in the admin area.
🛱 Plan	>		Assigned project runners		Group runners
Code	>		#2 (Pwo-zra1H) v1	P II Remove runner	These runners are shared across projects in this group.

Si queremos ver información sobre el Runner, miraremos el siguiente fichero -> /etc/gitlab-runner/config.toml Con ésto, ya tendríamos instalado nuestro Runner y conectado con Giitlab.



3.4. Configuraciones

Para la correcta configuración del clúster, tendremos que tener en cuenta las siguiente configuraciones que son:

- La propia configuración de este mismo, en la cuál abordaremos el mantenimiento de las aplicaciones y el rendimiento de estas mismas.
- La configuración y conexión entre de Grafana y Prometheus para poder tener una monitorización de nuestro cluster en cuanto a rendimiento y disponibilidad.
- La configuración de Gitlab y su respectivo Runner para realizar la integración continua de nuestra aplicación.

Dicho esto, comenzamos con las respectivas configuraciones.

3.4.1. Configuración del clúster

En este apartado, veremos los diferentes apartados que nos ofrece nuestro cluster, limitaremos el uso de nuestras aplicaciones y el uso del escalado de pods desde la interfaz gráfica.

Por ello, al acceder a nuestro cluster, podemos observar el uso de nuestra CPU, RAM y la cantidad de pods de nuestro nodo.

👻 Ranch	ner ×	🗿 Grafana	×	🖲 Prometheus Time Series 🔀 🏼	🦊 Administrator / mi-p	rimer× +				~	•	Naveg	ación pri	vada		×
$\leftarrow \rightarrow$	С	O 🗛 http	os://192.168.122.3		explorer#cluster-even						ជ					
≡	📙 cluster-k3s							Only User Namespa	ices 🗸	1 1	٤	ľ	٦	Q	:	2
^	Cluster Projects/Namespaces Nodes Cluster and Project Mer Events Workloads Apps Service Discovery Storage	nbers (H 0))))	Cluster Da Provider: K3s 170 Capacity	ashboard Kubernetes Version: v1295 48a1 Total Resources	Created: 3.9 hours ag	°	Node		11		¢ In: Deplo	stall Moni	itoring	Add Cl		
	roncy More Resources		Pods Used 14/1	10 V Scheduler V C	12.73%	CPU Reserved 0.2/4.cores		5.00%	Memory Reserved 0.14/3. Used 0/3.82 GiB	82 GiB					3.66%	





Como podemos observar, vemos que tenemos algunos apartados en los cuales se pueden ver proyectos o namespaces, estado de los nodos y los eventos sucedidos en este cluster.

Posteriormente en el apartado de los Workloads, podemos ver que podemos configurar nuestros despliegues de las aplicaciones con CronJobs, DaemonSets, Deployments, Jobs, StatefulSets e incluso ver todos los **pods** correspondientes.

💎 Ranch	er ×	🌀 Grafana	×	Prometheus Time Series ×	🖊 Administra	tor / mi-prime ×	+			~	😣 Navegaci	ón privada	- • ×
$\leftarrow \rightarrow$		O 🔒 http	s:// 192.168.122.3 :84			kload							: එ ≡
≡	📙 cluster-k3s							Only User Namespaces		٤ 🗅	ر ۵	с С	÷ 😂
	Cluster	>											
	Workloads	~	Workloads										Create
СКЗ										-			
' #'	DaemonSets		O Redeploy		Delete					=	Fi		
	Deployments	(+) 4											
		(4) 0	□ State ◯	Name 🖓		Туре⊖	Image 🗘		Restart	ts Age⊖	Healt		
	StatefulSets												
	Pods		Namespace: defau	lit									
	Apps					Deployment	grafana/grafana:11.0.0			3.7 hou	s 💻		••
	Service Discovery					StatefulSet	guav.jo/prometheus/alertmanager:v0.27.	0		3.7 hou			
	Storage												
	Policy					Deployment	registry.k8s.io/kube-state-metrics/kube-	state-metrics:v2.12.0		3.7 hou	s 💻		••
	More Resources					DaemonSet	quay.io/prometheus/node-exporter:v1.8.	0		3.7 hou	rs 📃	_	•• :
						Deployment	quay.io/prometheus/pushgateway:v1.8.0			3.7 hou	rs 📃		•• :
						Deployment	quay.io/prometheus-operator/prometheu +1more	us-config-reloader:v0.73.2		3.7 hou	rs 📃		••

A continuación, podemos ver tanto las aplicaciones con sus respectivos charts y repositorios, como todo el tema de servicios **Ingress**, **Services** y los **HorizontalPodAutoscalers**.

🐨 Ranch	er × 6	Grafana	×		rator / mi-prime × +				~		Navegac	tión privada	•	ø x
$\leftarrow \rightarrow$	c c	🔉 🔒 http	s:// 192.168.122.3 :8	443/dashboard/c/c-m-4hqm267j/explorer/se						☆				
=	📙 cluster-k3s						Only User Namespaces			È.	ĉ,	ρm		8
A	Cluster Workloads		Services 🕸										Cre	rate
СКЗ	Apps Service Discovery	> ~	± Download Y	AML 🗯 Delete					≡	•				
	HorizontalPodAutoscaler Ingresses	rs (Ho (Ho	State 🗘	Name 🗘	Target 🗘	Selector 🗘				Тур	•0	Age		
	Services Storage	(n) 10 >	Namespace: defa	ult										
	Policy More Resources				service 1¹ 3000/TCP	app.kuberne app.kuberne	tes.io/instance=grafana tes.io/name=grafana			Clu	ster IP	3.7 hou	rs	
					[Any Node]:31789	app.kuberne app.kuberne	tes.io/instance=grafana tes.io/name=grafana			Noc	le Port	3.7 hou	rs	
					https 7 ¹ 6443/TCP					Clu	ster IP	4.1 hou	rs	
					10.43.236.69.9093 <mark>1¹ http/TCP</mark>	app.kuberne app.kuberne	tes.io/instance=prometheus tes.io/name=alertmanager			Clu	ster IP	3.7 hou	rs	
 12					9093 <mark>1⁴ http/TCP</mark>	app.kuberne app.kuberne	tes.io/instance=prometheus tes.io/name=alertmanager			Hea	dless	3.7 hou	rs	
					http 1 ⁻¹ 8080/TCP	app.kuberne app.kuberne	tes.io/instance=prometheus tes.io/name=kube-state-metrics			Clu	ster IP	3.7 hou	rs	
*					10.43.16.136.9100 T¹ 9100/TCP	app.kuberne app.kuberne	tes.io/instance-prometheus tes.io/name=prometheus-node-exp	orter		Clu	ster IP	3.7 hou	rs	
•	Cluster Tools				10.43.72.147:9091 T 9091/TCP	app.kuberne app.kuberne	tes.io/instance-prometheus tes.io/name=prometheus-pushgate	way		Clu	ster IP	3.7 hou	rs	





Un dato importante es que la instalación de las aplicaciones la hemos realizado con Helm, por ello **los repositorios añadidos para nuestra instalación no se ven reflejados**. Estos mismos se encuentran en nuestro nodo master.

🐨 Ranch		🧔 Grafana	× 🥚 Prometheus Time S	ries 🛪 🤟 Administrator / mi-prime 🛪 🕂			🙁 Nav	egación pr	ivada 😑	ØX
$\leftarrow \rightarrow$		O 🔒 http:	s://192.168.122.3:8443/dashboard/c/c-m-4	hqm267j/apps/catalog.cattle.io.app			\$			ഹ ≡
≡	📙 cluster-k3s				Only User Namespaces 🗸 🛧		, ©	Q	œ :	8
•	Cluster									~
	Workloads		An installed application is a Helm 3 chart th	at was installed either via our charts or through the Helm CLI.						~
скз										
		(+) 2	🛓 Download YAML 📲 Delete			≡	•			
	Repositories									
	Recent Operations		State State Name State	Chart 🛇	Upgradable 🗘		Reso	urces A	ge 🗘	
	Service Discovery									
	Storage		Namespace: default							
	Policy			grafana:8.0.1				, 3.	8 hours	
	More Resources			prometheus:25.21.0			2	3 3.	8 hours	8

Un apartado también muy importante para el despliegue de nuestras aplicaciones es el almacenamiento o **Storage**. Este está compuesto lso **PersistentVolumes, StorageClasses, Configmaps** (donde se encuentran las variables de entorno), **PersistentVolumesClaims y Secrets** (donde se almacenan todas las contraseñas de manera encriptada) de nuestro clúster.

👻 Ranch	er ×	🧔 Grafana	×	Prometheus Time Series ×	🖊 Administrator / mi-prime ×	+			~		Nave <u>c</u>	jación pri	vada 📒	. ø x
$\leftarrow \ \rightarrow $		O 🔒 http	s:// 192.168.122.3 :8		j/explorer/persistentvolume					☆				മ ≡
≡	📙 cluster-k3s						Only User Namespaces				Ö	Q	ш	8
A	Cluster Workloads		Persistent	∕olumes ☆										Create
скз	Apps Service Discovery		⊥ Download Y	/AML										
	Storage PersistentVolumes	~ 2	□ State≎	Name 🗘		Reclaim Policy 🗘	Persistent Volume Claim 🗘		Source 🗘	F	Reason O	Ag	te 🗘	
	StorageClasses	1				Delete			Local			3.8	3 hours	
	ConfigMaps PersistentVolumeClai					Delete			Local			3.1	3 hours	:
	Policy More Resources													

También una opción muy importante puede ser la **parte de políticas**, **seguridad y monitoreo**. En estos podemos ver los límites de rangos, políticas de seguridad de las redes, cuotas de almacenamiento y los pods de reserva para los deployments expuestos.

Además en el apartado de monitoreo, podemos controlar las alertas, los tipos de monitorización y la configuración avanzada de nuestras máquinas de Grafana y Prometheus.





Por último, tenemos todos los demás recursos que nos ofrece Rancher que pueden ser APIs, aplicaciones, autoescalado, redes, nodos, organizador de los eventos, RBAC...

🐨 Ranch	er 🛛 × 🧔 C	Grafana	×	📀 Prometheus Time Series 🗵 🤟 Admini:	strator / mi-prime × +				~		🕺 Nave	egación	privada	•	e x
$\leftarrow \rightarrow$	с 0	🔏 http	s:// 192.168.122.3 :8							☆				٤	
≡	<u> c</u> luster-k3s						Only User Namespaces	Ţ	ک	•	()	Q	æ		3
ћ Скз	MutatingWebhookConfigurat ValidatingWebhookConfigura API APIServices		ClusterRole	eBindings ☆ AML ■ Delete								Filter	Create	from YAI	ML
	CustomResourceDefinitions FlowSchemas		State \$	Name 🗘	Role 0	Users 🗘	Groups 🗘	Service	Accounts	े tle			Age ≎		
	Apps				ClusterRole/cattle-fleet-system-fleet-agent-role			cattle-fle	eet-syster	m/fleet	-agent		4.2 hou		
	Nodes				ClusterRole/cattle-impersonation-u-613ryfqyeu			cattle-in imperso	npersonat nation-u-	tion-sy: 6l3ryfo	stem/cat iyeu	ttle-	4.2 hou		
	PersistentVolumes				ClusterRole/cattle-impersonation-u-wdxmpn3y4s			cattle-in imperso	npersonal nation-u-	tion-sy: wdxmp	stem/cat on3y4s	ttle-	4.2 hour		
	ConfigMaps Endpoints	€ 5 (+) 10			ClusterRole/cattle-impersonation-user-r62lc			cattle-in imperso	npersonat nation-us	tion-sy: er-r62	stem/cat lc	ttle-	4.2 hour		
	PersistentVolumeClaims				ClusterRole/cattle-unauthenticated		system:unauthenticated						4.2 hour		

Nota: RBAC es un mecanismo de control de acceso que define los roles y los privilegios para determinar si a un usuario se le debe dar acceso a un recurso.

Tras esta breve introducción a la interfaz de Rancher según nuestro clúster, pasamos a ver cómo se pueden limitar nuestros recursos de nuestros deployments y como podemos escalarlos de manera gráfica.

Para poder realizar estas 2 tareas, nos vamos al apartado de Workloads -> Deployments y dentro de este mismo podemos ver todos los despliegues de los distintos namespaces.

👻 Ranch	ner ×	orafana 🌀	×	👵 Prometheus Time Series 🗵	🦊 Admini:	istrator / mi-prime: × +					~	🕺 Nave	egación pr	ivada	- • ×
$\leftarrow \rightarrow$		O 🗛 http:	s:// 192.168.122.3 :8			apps.deployment					☆			. ⊲	
=	📙 cluster-k3s						Only User Na	mespaces		Δ.	× 🗈	¢	Q	æ	: 😂
	Cluster														
	Workloads		Deploymen	nts 🛱											Create
скз															
' #'	DaemonSets		C Redeploy							I	=	•			
	Deployments	₩4													
	Jobs		□ State ≎	Name 🗘		Image 🗘	Ready	ି Up To Date ି	Available 🗘	Restarts	Age 🗘		lealth		
	StatefulSets														
	Pods		Namespace: defa	ult											
	Apps					grafana/grafana:11.0.0					3.9 hour		_	_	
	Service Discovery					registry.k8s.io/kube-state-metrics/kube-state-metrics:v2.12.0	1/1				3.9 hour				
	Storage														
	Policy					quay.io/prometheus/pushgateway:v1.8.0					3.9 hour				
	More Resources					quay.io/prometheus-operator/prometheus-config-reloader:v0.73.2 + 1 more					3.9 hour		_		





Elegimos un deploy que queremos, que en mi caso es el grafana y podemos ver que escalamos los pods mediante la interfaz gráfica:

👻 Ranch	er ×	🧔 Grafana	× 📀 Prometheus Time Series 🛛 🤟 Administrator / mi-prime × 🕂				~	🙁 Naveg	gación privada	•	a x
$\leftarrow \rightarrow$		O 🖧 http	:// 192.168.122.3 :8443/dashboard/c/c-m-4hqm267j/explorer/apps.deployment/default/gr	afana#pods			☆				
≡	📙 cluster-k3s			Onl	ly User Namespaces		٤ 🗅	Ö	рш		C
п Скз	Cluster Workloads CronJobs DaemonSets Deployments Jobs	> (+) 0 (+) 1 (+) 4 (+) 0	Deployment: grafana (Active) Namespace:defail: Age:4hours: Pod Restarts:0 Image: grafana/grafana:11.00 Ready: 1/1 Up-to-date: 1 Available: 1 Endpoints: (AwnHodel33789 Labes:: Sapabatemetesio/imanaged.by:Helm appl.ubemetesio/imanaged.by:Helm appl.ubemetesio/imanage	grafana appikubernetesiolversion: 1100	helmshithart : grafana 80.1				Detail Co	onfig	
	StatefulSets Pods Apps Service Discovery Storage	(m) 1 (m) 6 > > >	Pods by State Image: Pods Services Ingresses Conditions Recent Events Related Resource						Scale		1 +
	Policy More Resources		Download VAML Delete State Name Image grafana-64466475d-217ht grafana/grafana:	Ready \(\) 1100 1/1	ි Restarts ි 0	IP 0 10.42.0.20	No	de 🗘 ster	Age \Diamond 4 hours		

Si queremos limitar el uso de recursos de un deployment, le damos a los 3 puntos y le damos a Edit Config. Tras esto, en el mismo servicio grafana, le damos a Resources y limitamos los recursos como queramos.

🐨 Ranch	er ×	🧔 Grafana	× . 🥚 Prome	theus Time Series 🔀 🦊 A	Administrator / mi-prime >	× +					~	•	Naveg	jación pr	ivada) ×
$\leftarrow \rightarrow$		O 🖧 http	s:// 192.168.122.3 :8443/dashb									☆				ව £	
=	📙 cluster-k3s							Only	ly User Namespaces	↥		•		Q	æ		3
СКЗ	Cluster Worklaads CronJobs DaemonSetts Jobs StatefulSets Pods Apps Service Discovery Storage	>	Deployment: grafa Namespace default Age: 41 This Deployment is managed default Replicas* 1	ana Active Nours Pod Restarts: 0 by the Helm app grafana; char	nges made here will likely be Name * grafana	overwritten the next time H	Helm runs. Description Any text										
4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Policy More Resources		General Health Check Resources Security Context Storage	Resources CPU Reservation e.g. 1000 CPU Limit e.g. 1000 NV/IDLA CPU Limit/Reserve			mCPUs mCPUs mCPUs								<u>.</u>		

Tras hacer esto en algunos otros deployments con alta carga de trabajo y recursos. Tras un tiempo de espera para la reestructuración de los pods, tendremos el cluster limitado sin la utilización de todos los recursos.





3.4.2. Configuración de métricas con Grafana + Prometheus

Pasamos a la configuración y conexión entre Grafana y Prometheus. Como tenemos expuesto los servicios mediante **NodePort** podemos ver los servicios creados.

Para la conexión entre las máquinas de **Grafana** y **Prometheus**, tendremos que añadir una nueva conexión mediante la interfaz gráfica.

Por ello, accederemos a Grafana y podemos ver que tiene distintos apartados como los **Dashboards de las métricas**, las **alertas** de notificación mediante un límite de cuotas o reglas, las diferentes conexiones para añadir como **Azure**, **AWS**, **GitHub**, **GitLab**, **Oracle**, **PostreSQL**...



En el mismo apartado de conexiones, buscamos Prometheus y clicamos en él para añadir una nueva conexión.

÷		0 👌 🕶	192.168.122.31:31789/connections/add-new-connection		
Ø					😅 ctrl+k
≡	Home > Connections >				
> >	⑥ Home ☆ Starred 昭 Dashboards ❷ Explore		Add new connection Browse and create new connections		
,			Q promethe		
*	 Connections Add new connection 		😫 Data sources		
>	Data sources Ø Administration		Alertmanager	Prometheus	Prometheus AlertManager Datasource





Cuando accedemos, tendremos que ingresar la URL de nuestro servicio de Prometheus con el respectivo NodePort, ya que se encuentran en la misma máquina:

$\leftarrow \rightarrow$	୦ <u>୫</u> ୦	192.168.122.31:31789/connections/datasources/edit/	dojrpv6o4ttse						
Ø			Q Search or jump to						
= н	lome > Connections > Data sources								
G → ☆ Ⅲ → ② → ④ →	Home Starred Dashboards Explore Alerting Connections	prometheus Type: Prometheus <u>NH Settings</u> E Dashboards	prometheus Type: Prometheus HF Settings III Dashboards						
	Add new connection Data sources	hnection Configure your Prometheus data source below Cor skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hosted data sources from Grafa							
, ,		Name O prometheus Before you can use the Prometheus data source, you n Fields marked with * are required	Default		iew the documentation.				
		Connection Prometheus server URL * O http://192.168.122.31:321	95						

Tras esto, bajamos hasta donde pone Save and Test y guardamos la conexión con nuestra máquina Prometheus.

Con esto comentado, pasamos a crear el Dashboard con las métricas respectivas para hacer ver una interfaz bonita para el monitoreo de métricas. Por ello, nos vamos a este apartado y creamos un nuevo Dashboard y añadimos una nueva visualización:

Ø		Q Search or jump to Ctrl+k	+~ 💿 🔉 👯
= н	ome > Dashboards		^
。 、 ☆ 、 出	Home Starred Dashboards	Dashboards Create and manage dashboards to visualize your data	New ^
9 > 4 > 0		Q. Search for dashboards and folders ▷ Filter by tag → □ Starred	New folder Import
> @		Name 88 Alertmanager / Overview 88 CoreDNS 88 etcd	Tags alertmanager-mixlin coredns dns etcd-mixlin





Ø			Q Search or jump to	📟 ctrl+k					H	+ ~	?)
= 1	Home > Dashboards > N						Add ~	② Last 6 hours		Q	G	
6												
~ 8	B Dashboards											
			Start your new o	lashboard by a	ddin	n a	visua	lization				
		_	otart your new t		aam	y u	viouu					
			elect a data source and then qu	ery and visualize your data w markdowns and other wid	ith char aets.		ts and table	es or create lists,				
÷Ę				Add visualization	on							
ଢ					on-							
<u>ن</u>												

Seguido de esto, apuntamos a nuestro Prometheus por defecto ya viene configurado en nuestro stack de Helm.



Cuando accedemos, tendremos la interfaz la cual podremos configurar las métricas de Prometheus mediante una interfaz gráfica.

Ø		Q Search or jump to C ctrl+k	+~ 💿 🔉 👹
≡ Ho	ome > Dashboards > New	shboard > Edit panel	Discard Save Apply
		Table view • Fill Actual O Last 6 hours • Q C	Time series
~ 🔡	Dashboards Playlists	Panel Title	Q Search options All Overrides
			✓ Panel options Title Panel Title
		🔋 Query 🧻 👔 Transform data 💿 🔒 Alert 💿	Description
		Data source Prometheus v O > Q. MD = auto = 707 Interval = 30s Query inspector	Transparent background
			Panel linksRepeat options
		Metric Label filters Select metric Select label + Operations > Options Legend: Auto Format: Time series Step: auto Type: Range Exemplars: false	 Tooltip Tootip mode Single All Hidden Hover proximity How close the cursor must be to a point to trigger the tooltip, in pixels







Algunas de las métricas están relacionadas con el uso de CPU, el consumo de bytes para los procesos, el almacenamiento de nuestro cluster...

También en el apartado Dashboards, podemos importar Dashboards con información referente a nuestro cluster de manera más rápida y sencilla.

Para esto, accederemos a estos mismos y le damos a New -> Import e importamos la plantilla que queramos.

								1
С	0 🔒 🛛	+ 192.168.122.31:31789/dashboards			☆☆			
								a (
ome > Dashboards								
Home Starred		Dashboards					Nev	v ^
Dashboards							ew das	hboard
Playlists							ew fold	
Snapshots		Starred		C			nport	
Library panels								
Public dashboards								
Explore			No dashboards yet. Create your first!					
Alerting								
Connections			+ Create Dashboard					
Add new connection								
Data sources								

Para poder importar un Dashboard correctamente, indicamos un ID un import mediante un formato JSON de alguien de la comunidad que tenga métricas para nuestro cluster de K3S.







Un ejemplo puede ser 19972, el cual hace referencia al cómputo de recursos que tiene nuestro cluster de K3s. Cuando le demos a Load, nos pedirá que indiquemos el **resource** donde obtendremos las métricas (el cual, por defecto, es la conexión Prometheus de nuestro cluster).

C	0 🔒 어	192.168.122.31:31789/dashboard/import	
		Q S	earch or jump to
ome > Dashboards >			
Home Starred Dashboards Playlists Snapshots Library panels Public dashboards Explore Alerting		Import dashboard Import dashboard from file or Grafana.com Importing dashboard from Grafana.com Published by Updated on Options	
Connections Add new connection Data sources Administration		Name K3S Monitoring Folder Dashboards Unique Identifier (UID) The unique Identifier (UID) of a dashboard can be used for uniquely Identifi dashboard between multiple Grafana Installs. The UID allows having consis URLs for accessing dashboards or changing the title of a dashboard will no any bookmarked links to that dashboard. IbI4P-xiz Prometheus Frometheus The Definition of the Cancel Import Cancel	v a tent t break Change uid

Para finalizar, lo importamos y accedemos a este mismo mediante Dashboards y veremos que tiene todo tipo de información sobre nuestro cluster.





	G 0 6 0							
	rtar marcadores 🖕 Comenzar a i	Jsar Firefox						
			O. Search or jump to		atel i k		+ u	<u>ه</u> م
			Q Search of Jump to					
here and a set of the	ome > Dashboards > K3S Monito	ring 🟠					a c	
Since Debaoes Note Note </td <td></td> <td>~ Pods CPU usage</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>		~ Pods CPU usage						
Determined Image: Construction		Dede CDI Lucase (Em aug)						
Another is a set of the set of	Deetheesde	Pous CPO usage (Sill avg)						
Payloth	Dashboards							
Sequence for the sequence of t	Playlists	0.800%				 cattle-cluster-agent-84c865f7d7-ctgk2 		
Likery panel. Partice databases Partice databases						 coredns-6799fbcd5-szkjr 		
Advecture constrained of the set		g 0.600%				 fleet-agent-f7dc57db7-fbvlk 	0.074%	0.0655
Pindle databases For a possible databases For a possible databases Constant Con						 grafana-6446f6475d-2f7ht 		0.1569
Export Ammy Creation Addresser Creation De Laces Administration De Laces Admin	Public dashboards					 local-path-provisioner-6c86858495-cxtsx 	0.008%	0.0119
		0.200%				 metrics-server-54fd9b65b-smpkx 	0.281%	0.2169
Contractions Add near contractions Data counters Add near contractions Data counters Add near contractions Add near contractions A		0.000%				prometheus-alertmanager-0	0.086%	0.0963
Canada		12-30 12-35				prometneus-kube-state-metrics-1/C68b841-89/6k	0.045%	
Add macrotaction Dela sources Administration Administ		 Containers CPU usage 						
definition of the set		Containers CPU usage (5m avg)						
Home Survey Surve								
Here Survey locate				- doc	cker: docker.io/grafana/grafana:1	11.0.0 (708207c50bf48b6424e4802e818733595e99bc7f511a99ff.		
Or Control Con		0.80%		doc	cker: docker.io/rancher/fleet-age	nt:v0.9.5 (54bd9afa77dac7ec5ea5ace90a26d9afc67d45383a7a		0.06
		g 0.60% V		V _ doc	cker: docker.io/rancher/klipper-lb	0:v0.4.7 (b6b9dcaaf843786277ff93b3aad1ff1006bc14e7d66a050		
Decision and a constraint of the constra		5 0.40%		— doo	cker: docker.io/rancher/klipper-lb	0:v0.4.7 (f33933c1e125780057b0406932a2b6fbdd441c318ee120		
				- doc	cker: docker.io/rancher/local-path	h-provisioner:v0.0.26 (b2a523ee23b0fba460062d303db886a76		
		0.20%		- doo	cker: docker.io/rancher/mirrored-	-coredns-coredns:1.10.1 (affc7da93b8e4fd0bb313e7c3e4c64af9		
123 123 124 124 125 edder edder <td></td> <td>0.00%</td> <td></td> <td>- doo</td> <td></td> <td>library-traefik:2.10.7 (6da63bcb31ad6f7143bb509cb3c69a91767.</td> <td></td> <td></td>		0.00%		- doo		library-traefik:2.10.7 (6da63bcb31ad6f7143bb509cb3c69a91767.		
Al processo CPU usage Al processo CPU usage Al processo CPU usage Total Total<		12:30 12:35	12:40 12:45 12:5	0 12:55 — doo	cker: docker.io/rancher/mirrored-	-metrics-server:v0.7.0 (09beee3ef81491504c2606ce053643891f	0.28%	
All processes CPU usage fam any store ar narradones. Concertors a usaf lifedo ar parabolands - K3S Monitoring sarradones. Concertors a usaf lifedo Dashbaards - K3S Monitoring Concertors a usaf lifedo Dashbaards - K3S Monitor		~ All processes CPU usage						
All processor (2) usage (5m arg) 1007 C 0 0 0 192.164.1223.1128/d/(164/h-%//32)=monitoring/org/d=164refreil=1m ar nariadores. C 0 192.164.1223.1128/d/(164/h-%//32)=monitoring/org/d=164refreil=1m C 0 search or jump to C 0 search or jump to		. An processes of a dadge						
C O O I ISSUE C O C O I ISSUE C O I I ISSUE C O I I I I I I I I I I I I I I I I I I		All processes CPU usage (5m avg)						
C C		All processes CPU usage (5m avg)						
Contractor Contractor <td></td> <td>All processes CPU usage (5m avg) 10.00%</td> <td></td> <td></td> <td></td> <td></td> <td>1</td> <td></td>		All processes CPU usage (5m avg) 10.00%					1	
ar narcadore, Containers memory usage Contain	a oa.	All processes CPU usage (5m avg) 1000%	n²ornid=1&reFresh=1m			λ φ	Ø	ა <i>s</i> h ≡
Q Sarch or jump 0 Im ch	с <u>о</u> во	All processes CPU usage (5m avg) 10.00% 192.166.122.31:31789/d/1bi4P-xiz/k3s-monitoring	j?orgld=1&refresh=1m			ža (2	0) එ =
where > Dashbeards > K3S Montors * C Max Max <td>ල 〇 은 ං tar marcadores 🍁 Comenzar a L</td> <td>All processes CPU usage (5m avg) 10.00% * 192.168.122.31:31789/d/1bi4P-xiz/k3s-monitoring sar Firefox</td> <td>j?orgid=1&refresh=1m</td> <td></td> <td></td> <td>滿 合</td> <td>0</td> <td>) ඩ =</td>	ල 〇 은 ං tar marcadores 🍁 Comenzar a L	All processes CPU usage (5m avg) 10.00% * 192.168.122.31:31789/d/1bi4P-xiz/k3s-monitoring sar Firefox	j?orgid=1&refresh=1m			滿 合	0) ඩ =
Name	C O 권 o tar marcadores 🎽 Comenzar a u	All processes CPU usage (Sm avg) 10.00% • 192.168.122.31:31789/d/1bi4P-xiz/k3s-monitoring saf Firefox	g?orgid=1&refresh=1m Q. Search or jump to	5 ct	rl+k	及 合		ා ඩ ≡) ක (į
Home Control None	ල O 은 o tar marcadores 🎽 Comenzar a U	All processes CPU usage (5m avg) 10.00% • 192.168.122.31:31789/d/1bl4P-xiz/k3s-monitoring sar Firefox	g?orgid=1&refresh=1m Q. Search or jump to	in ct	лы	× Ω	♥ () + ~ ()	ා ඩ = _ බ (
Nume Display Standd Pods memory usage i Dasboards Pods memory usage i Playldts Stapphots i <	ල Q 관 ං tar marcadores,, 🦛 Comenzar a u ime > Dashboards > K3S Monitor	All processes CPU usage (5m avg) 10.00% • 192.168.122.31:31789/d/1bi/4P-xiz/k3s-monitoring sar Firefox	ŋ?orgid=1&refresh=1m Q. Search or jump to	e ct	л+к	え 分 ③ Add マ 50mm ① Last 30 minutes マ 〇	- + - - 3 1⊓	ອ £] ≡
Stared Dashoods Pipelydata Snapshots Library panels Public dashoards Explore Aderting Connections Add new connection Data sources Administration Data sources Data source	C O A o tar marcadores In Comenzar a u ume → Dashboards → K3S Monito	All processes CPU usage (5m avg) 10.00%. • 192.168.122.31:31789/d/1bi4P-xiz/k3s-monitoring saar Filefox	g?orgid=1&refresh=1m Q. Search or jump to		n+k	🕅 क्रि () Add v Share () Last 30 minutes v ()	+ ~ ©	ວ £1 ≡ 0 ລ € 1 ~ _ ^
Pois memory usage Nm	C O 관 o tar marcadores I Comenzar a u ome > Dashboards > K3S Monitor Home []	All processes CPU usage (5m avg) 10.00% • 192.168.122.31:31789/d/1bl4P-xiz/k3s-monitoring sar Firefox - Pods memory usage	g?orgid=1&refresh=1m Q. Search or jump to	ca ct	nek B	⊼A Ω	ی کی اس ۱۳	ຍີ ຄື ≣ ເຊິ່ງ ຄັ້ນ ເຊິ່ງ ຄັ້ນ ເຊິ່ງ
Pixelist Name Maar List* Snaphots 250 MB - 2010-102 (2010) 2010 MB	C C Comencar a u tar marcadores Comencar a u ome > Dashboards > K3S Moniton Home ID Starred	All processes CPU usage (Sm avg) 10.00% • 192.168.122.31:31789/d/1bi4P-xiz/k3s-monikoring sar Firefox • Pods memory usage	prorgid=1&refresh=1m Q. Search or jump to		n+k ©	ইন ট্র ি Add v Share ে C Last 30 minutes v C	ی چ + × (۵ ک ۱۳	ම ඩ ≡ බ ද බ × ^
Amplada 256 MB 256 MB 207 MB 209 MB 200 MB	C O A o tar marcadores M Comenzar a u ome > Dashboards > K3S Monitor Home II Starred Dashboards	All processes CPU usage (5m avg) to cotk ■ 192.168.122.31:31789/d/1b/4P-xiz/k3s-monitoring sar Firefox Ing ☆ ■ Pods memory usage Pods memory usage	g7orgid=1&refresh=1m Q. Search or jump to	en etr	nex B	र्दे क्षेत्र ि Add v Start © Last 30 minutes v C	ی اور ۱۳ ۱۳) £] ≡) ,
Shaphots Low Ma Low Ma Low Ma 25.5 MB 26.5 MB 20.5 MB	C C 2 c tar marcadores I Comenzar a ome > Dashboards > K3S Monitor Home II Starred Dashboards	All processes CPU usage (5m avg) 10.00% • 192.168.122.31:31789/d/1bi4P-xtz/K3s-monitoring sar Firefox • Pods memory usage Pods memory usage	g?orgid=1&refresh=1m Q. Search or jump to	. er	ntak B	م کر کر ا	ک و + + ۲ ۵ ک ۱۳) £) ≡)
Library panels 102 MB - fmet-squer:7nd:53/20-r/truk 36.9 MB 38.1 MB 38.1 MB 38.1 MB 39.1 MB 30.1 MB 20.2 MB - metrics-struct-5/dblbbbs-mpix 21.9 MB 2.0 MB 7.0 MB <t< td=""><td>C C A contained of the second of the second</td><td>All processes CPU usage (5m avg) 10.00%</td><td>g?orgid=1&refresh=1m Q. Search or jump to</td><td></td><td>nak B</td><td>Addi v Share O Last 30 minutes v O Name cattle-cluster-agent: 84:08857/37-rtgk2</td><td>♥ @ + v © 0.011 0.011 Mean 207 MIB</td><td>● 2 = →</td></t<>	C C A contained of the second	All processes CPU usage (5m avg) 10.00%	g?orgid=1&refresh=1m Q. Search or jump to		nak B	Addi v Share O Last 30 minutes v O Name cattle-cluster-agent: 84:08857/37-rtgk2	♥ @ + v © 0.011 0.011 Mean 207 MIB	● 2 = →
Public dashbards Explore Alerting Connections Add new connection Data sources Administration Administra	C C A o tar marcadores A Comenzar a u me > Dashboards > K3S Monitor Home I Starred Dashboards Playlists Snapshots	All processes CPU usage (5m avg) to cotx. 192.168.122.31:31789/d/1b/4P-xiz/k3s-monitoring sar Firefox ing ☆ - Pods memory usage Pods memory usage 250 MB	g7orgid=1&refresh=1m Q. Search or jump to	- • • • • • • • •	nex	Xn ☆ ③ Add ✓ Source ③ Last 30 minutes ✓ € Name ● catilit-cluster-agent-84c8857/37-ctgk2 ← carcein-6999hcbd5=x28/r	♥ € + - ↓ © © 1n 0.011 Mean 207 MiB 26.5 MiB	209 MIB 26.9 MIB
Add new connection 6 4 MB 7 2 MB 8 2 MB 7 2 MB	C ○ 2 o tar marcadores I Comenzar a me → Dashboards → K3S Monitol Home ED Starred Dashboards Playlists Snapshots Library panels	All processes CPU usage (5m avg) 10.00%	prorgid=1&refresh=1m Q. Search or jump to		¹¹⁴	Name - cattle-cluster-agent-84c885/747-ctgk2		20 €1 =
Explore 64 Mg - metrics-strutter	C C Comencar a un tar marcadores Comencar a un ome > Dashboards > K3S Moniton Home C Starred Dashboards Playlists Snapshots Library panels Public dashboards	All processes CPU usage (Sm avg) 10.00% • 192.168.122.31:31789/d/1bi4P-xiz/k3s-monitoring saar Fikefox • Pods memory usage Pods memory usage 256.M8 • 192.M8 • 193.M8 • 19	g?orgid=1&refresh=1m Q. Search or jump to		лна В	Name C Last 30 minutes C e catile-cluster-sgent-94c685/747-ctgk2 coredns-6709fbcc5-szkjt cester-sgent-94c685/747-ctgk2 g ordans-6709fbcc5-szkjt godara-64468/2592-z7twix godara-64468/2592-z7twix		209 MB 22.1 MB 92.1 MB
Alerting Obs	C C C C C C C C C C C C C C C C C C C	All processes CPU usage (5m avg) 10.00%	g?orgid=1&refresh=1m Q. Search or jump to			Xn ☆ Image: Source Image: Source Name Image: Source = cattler-cluster-agent: 8668557/07-ctgk2 = cattler-cluster-agent: 8668557/07-ctgk2 = cattler-cluster-agent: 76c55/007-fbv/k = gatana=646f647362-7bv/k = gatana=646f647362-7bv/k = tocat-agent: 76c55/007-fbv/k = tocat-agent: 76c55/007-fbv/k		20 = = 20 20 209 MIB 269 MIB 38.1 MIB 92.1 MIB 10.8 MIB
Connections 1230 1230 1240 1245 1250 primals-cades/s/36/2/1M @ 1/2/M@ _ prometaus-kube-state-metrics-7/6888/41-807/k 12.3 MB	C C C Comenzar au tar marcadores Comenzar au tar marcadores Comenzar au tar marcadores Comenzar au tar marcadores Kashboards tarred Dashboards Palyists Snapshots Lubrary panels Public dashboards Explore	All processes CPU usage (Sm avg) 10.00%	g?orgid=1&refresh=1m Q. Search or jump to		nak B	Name cattle-cluster-agent 86c885707-ctgk2 cattle-cluster-agent 86c885707-ctgk2 cattle-cluster-agent 86c885707-ctgk2 infect-agent 76c57007-ftwik grafana-6446f64755-277hr infect-agent 76c57007-ftwik grafana-6446f64755-277hr infect-agent 96c58583495-cxtsx metrics-serve-64086853495-cxtsx		209 KB 38.1 MB 92.1 MB 22.2 MB 22.2 MB
Add new connection - Containers memory usage Data sources Containers memory usage - Con	C C Comencar a un tar marcadores Comencar a un Home C Starred C Dashboards Playlists Snapshots Library panels Public dashboards Explore Alerting	All processes CPU usage (Sm avg) 10:20% * 192.168.122.31:317/89/d/1bi4P-xiz/k3s-monitoring saar Firefox * Pods memory usage 256 MB 192 MB 4 64 MB 64 MB	g7orgid=1&refresh=1m Q. Search or jump to		7H4 	Xa ☆ Image: State - Cluster - Sigent - BaceBS7/07-ctgb2 Correlts - Cluster - Sigent - BaceBS7/07-ctgb2 Correlts - Correlts - Cluster - Sigent - BaceBS7/07-tcgb2 Correlts - Correlts - Cluster - Sigent - BaceBS7/07-tcgb2 Correlts - Correlts - Cluster - Sigent - BaceBS7/07-tcgb2 Correlts - Correlts - Cluster - Sigent - BaceBS7/07-tcgb2 Image: Sigent - BaceBS7/07-tcgb2 Image: Sigent - Sigent - BaceBS8/07-tcgb2 Image: Sigent - Sigent - BaceBS8/07-tcgb2 Image: Sigent - Sigent - Sigent - Sigent - Sigent - Cluster - Sigent - Sigent - Sigent - Sigent - Sigent - Sigent - Cluster - Sigent - Sige		209 MB 229.9 MB 229.9 MB 221.4 MB 222.2 MB 220.4 MB
Add mer Sameroon Container's memory usage C	C C 2 c car marcadores Comenzar a me -> Dashboards -> K3S Monitor Home I Starred Dashboards Playlists Snapshots Library panels Public dashboards Explore Alerting Commections	All processes CPU usage (5m avg) 10.00%	prorgid=1&refresh=1m		714 k B 1 1 1 1 2 12 2 5 2 5 5 5 5 2 7 11 1 1 1 2 12 5 2 5 5 1 1 1 1 1 1 1 1 1	Xa ☆ Image: State of the state		20 £) = 20 0 € 20 0
Data sources Containers memory usage Administration 256 MB 56 MB 6 doter: Idoter.iol/graf.au/g	C	All processes CPU usage (5m avg) 10.00%	37orgid=18xefresh=1m Q Search or jump to 	c ct c ct c ct c ct c ct c ct c ct c ct	71-k	Name • cattle-cluster-sgent: 846-8857/07-ctgk2 • cattle-cluster-sgent: 846-8858/8459-ctgk2 • promotheus-slattmanager. 0 • promotheus-slattmanager. 0 • promotheus-slattmanager. 0		209 MIB 26.9 MIB 26.9 MIB 38.1 MIB 92.1 MIB 10.8 MIB 22.2 MIB 20.4 MIB 12.6 MIB
Administration Administration Name Name Last* 256 MB - <td>C C Consections Armarcadores Consentar a Consentar as Consentar as Consentar as Consections Add new connection</td> <td>All processes CPU usage (5m avg) 10.00%</td> <td>27orgid=1&refresh=1m Q. Search or jump to</td> <td>c cb c cb c</td> <td>714 C</td> <td>Name ○ Last 30 minutes ∨ C • Cattle-Cluster-agent: 84c885707-ctgk2 • Cattle-Cluster-agent: 84c885707-ctgk2 • Cattle-Cluster-agent: 94c8858495-ctgk2 • Cattle-Agent: 94c8858495-ctgk2 • Inter-server-54d19865b-mgkk • promethous-subre-state-metrics-70c888841-8970k</td> <td></td> <td></td>	C C Consections Armarcadores Consentar a Consentar as Consentar as Consentar as Consections Add new connection	All processes CPU usage (5m avg) 10.00%	27orgid=1&refresh=1m Q. Search or jump to	c cb c	714 C	Name ○ Last 30 minutes ∨ C • Cattle-Cluster-agent: 84c885707-ctgk2 • Cattle-Cluster-agent: 84c885707-ctgk2 • Cattle-Cluster-agent: 94c8858495-ctgk2 • Cattle-Agent: 94c8858495-ctgk2 • Inter-server-54d19865b-mgkk • promethous-subre-state-metrics-70c888841-8970k		
Value Main Last 256 MB	C	All processes CPU usage (5m avg) 10.00%	porgid=1&refresh=1m Q. Search or jump to	c ct	171-K	Add v Share O Last 30 minutes v C Name catile-cluster-agent-86c885703*-ctgk2 condrais-67997bc03*-xtgi C		 20 £1 = 20 £1 = 209 MB 240 MB 92.1 MB 92.1 MB 12.6 MB 12.6 MB
200 mm	C C C A comenzar a ar marcadores C Comenzar a me > Dashboards > K3S Monitor Home E Starred Dashboards E Playlists Shapshots Library panels Public dashboards Explore Alerting Connections Add new connection Data sources Administration	All processes CPU usage (5m avg) 10.00%	g7orgid=1&refresh=1m Q. Search or jump to	c ctr	71+k © 12 12 25 959 22 26 959 24 6446(14756-2277ht B0.2 MIB	Name cordsh-5/09/ncc5-tztyl cordsh-5/09/ncc5-tztyl setsarpath-64869/39/27-tdg/2 cordsh-5/09/ncc5-tztyl setsarpath-64869/39/27-tdg/2 pdraft-64869/39/27-tdg/2 setsarpath-62869/39/27-tdg/2		209 MB 220 MB 222 MB 222 MB 224 MB
192 M8	C C C A ar marcadores Image: Commencer and Commencer an	All processes CPU usage (5m avg) 10.00%	2 ⁷ orgid=1&refresh=1m Q. Search or jump to	cii ctr cii ct	risk 113 k 12 12 50 50 12 12 50 50 14 4 606476475477771 80.2 MB	Name 		20 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
128 MB	C Q L ar marcadores ♦ Comenzar a me > Dashboards > K3S Moniton Home Bashboards Playlists Snapshots Library panels Public dashboards Explore Alerting Connections Add new connection Data sources Administration	All processes CPU usage (5m avg) 10.00%	37orgid=1&refresh=1m Q Search or jump to 	c ct c ct c ct c ct c ct c ct c ct c ct	71-k	Add v Start C Last 30 minutes v C Name catal-cluster-sgent-864885747-ctgk2 C		209 MB 209 MB 269 MB 269 MB 269 MB 269 MB 269 MB 269 MB 222 MB 10.8 MB 10.8 MB 12.6 MB 12.6 MB 12.6 MB
1/28 M0 - docker.j docker.jmanch/inspector.vov / 12393.11 in 1/25 vov / 12390.2142/c01003442.1201003442.12010103442.1201401034442.1201403444444444444444444444444444444444	C Q A ar marcadores Image: Commercer at a commercer a	All processes CPU usage (5m avg) 10.00%	27orgid=1&refresh=1m Q. Search or jump to	c ct c	1144 E	Name C Last 30 minutes C Name C Last 30 minutes C Cordin-Cluster-agent-84:685707-ctgb2 C Cordin-Colore-Col		209 MB 92 1 10 2 209 MB 921 MB 921 MB 922 AMB 922 AMB 922 AMB 922 AMB 923 AMB 923 AMB 923 AMB 923 AMB 924 AMB 924 AMB 925 AMB 925 AMB 926 AMB 926 AMB 927 AMB 927 AMB 927 AMB 928 AMB
64 MB 0 g = docker docker/aname/minuter/minuter/set (22) see 130/14/2002/23/24/23/04/23/04/23/24/23/04/23/04/23/24/23/04/23/04/23/24/23/04/23/04/23/24/23/04/23/04/23/24/23/04/23	C ○ 2 ○ ar marcadores Image: Comencar a comencomencar a comencar a comenco comencar a	All processes CPU usage (5m avg) 10.00%	porgid=1&refresh=1m Q. Search or jump to 12.40 12.45	c ch 2024-06-1 12:50 Nam 4 dock 4 dock 4 dock 4 dock	11-1 (2) 12 12 50-50 12 12 50-50 14 2 12 50-50 14 2 12 50-50 15 2 12 50-50 16 2 12 12 12 12 12 12 12 12 12 12 12 12 1	Name Consister - agent: 86:6857/07-ctgk2 e. cattler-cluster - agent: 86:6857/07-ctgk2 Consider: 6-097bc05-xtgk2 e. crosteris - 67:997bc05-xtgk2 Consider: 6-097bc05-xtgk2 e. crosteris - 6-097bc05-xtgk2 Consider: 6-097bc05-xtgk2 e. crosteris - 6-097bc05-xtgk2 Consider: 6-097bc05-xtgk2 e. prometheus - allert manager-0 Deprometheus - kube-state-metrics-77:698:084f-997bc05-xtgk2 v: 0.9.1 (54bc0481b7/70s-7/sc-5ke35.sc=002.02/03/scf67/d55383.7% A.2 (54bb941770s-7/sc=5ke35.sc=002.02/03/scf67/d55383.7% v: 0.9.1 (54bc0481770sc-7/sc=5ke35.sc=002.02/03/scf67/d55383.7% A.2 (74b97)-74/sc=7/sc=5ke35.sc=002.02/03/scf67/d55383.7% v: 0.9.1 (54bc0481780577mac-7xc=5ke35.sc=002.02/03/scf67/d55383.7% A.2 (74b97)-74/sc=7/sc=5ke35.sc=002.02/03/sc=004.647/d65.050 v: 0.9.1 (54bc0481780577mac-7xc=5ke35.sc=002.02/03/sc=004.647/d65.050 A.2 (74b67)-74/sc=7/sc=7/sc=004.7%	Image: Control of the second	200 MB 222 MB 224 MB
OB Control of the second	C Q A ar marcadores Comenzar au me → Dashboards → K3S Moniton Home Ime	All processes CPU usage (5m avg) 10.00%	37orgid=1&refresh=1m Q. Search or jump to L.40 L.42 L.42 L.42 L.42 L.42 L.42 L.42 L.42	c ctr	11+k Image: Compare the second s	Name C cation C Last 30 minutes C cation C Last 30 minutes C cation C Last 30 minutes C cation C coredns 67097bcc6-szkir edata partar gafara 64469/359:27ht edata partar partar 64469/359:27ht enertics partar partar 64469/359:27ht enertics partar partar 64469/359:27ht enertics partar partar 64469/359:27ht enertics partar and transference 0.0 0.0 (708207):500/48b6424e4802a8187333565690bc; 7/511a00f 0.0 (708207):500/48b6424e4802a8187333505690bc; 7/514b00f 0.0 (708207):500/48b6424e4802a8187333505690bc; 7/514b00f 0.0 (708207):500/48b6424e4802a8187333505690bc; 7/514b00f 0.0 (708207):500/48b6424e4802a8187333505690bc; 7/514b00f 0.0		200 MB 200 MB 20
6 8	C O A car marcadores Comenzar a comenzar a comenz	All processes CPU usage (5m avg) 10.00%	27orgid=1&refresh=1m Q. Search or jump to	C ct c	114 k 212 250 50 24 250 50 26 400 407/05 - 277hr 80 2 MIB 26 2 60 50 27 2 7 10 10 2 MIB 28 2 60 50 29 4 60 400 / 91	Name • Cast 30 minutes • C • Add • • • • • • • • • • • • • • • • •	C C C	
100 000 000000000000000000000000000000	C C C C C C C C C C C C C C	All processes CPU usage (5m avg) 10.00%	gTorgid=1&refresh=tm Q. Search or jump to Q. Search or jump to	c dr 2222-0-1 12:50 2222-0-1 docx - docx - docx - docx - docx - docx - docx - docx	11 k E 12 12 50 50 E 10 10 10 10 10 10 10 10 10 10 10 10 10 1	Name C Last 30 minutes C e catlis-cluster-agent-862885707-ctgk2 C C C e catlis-cluster-agent-862885707-ctgk2 C C C C e catlis-cluster-agent-862885707-ctgk2 C	Image: Constraint of the second sec	Last - 200 MB 22.0 MB 22.0 MB 22.0 MB 22.1 MB 22.1 MB 22.2 MB 22.2 MB 22.2 MB 22.2 MB 22.2 MB 22.2 MB 22.2 MB 22.8 KB 22.8 KB 23.7 P MB 28.8 KB 28.8 KB 28.8 KB 27.8 KB 28.8 KB 27.8 KB 27.7 KB 27.8 KB 27.7 KB 27.8 KB 27.7 KB 27.8 KB 27.7 KB 27.8 KB 27.7 KB 27.8 KB 27.8 KB 27.8 KB 27.7 KB 27.8 KB 27.
	C Q A tar marcadores Image: Comencar a target tar marcadores Image: Comencar a target Home Image: Comencar a target Home Image: Comencar a target Dashboards Image: Comencar a target Paylists Image: Comencar a target Shapshots Image: Comencar a target Library panels Image: Comencar a target Aderting Connections Add new connection Image: Comencar a target Data sources Image: Comencar a target Administration Image: Comencar a target	All processes CPU usage (5m avg) 10.00%	270rgid=1&refresh=1m Q. Search or jump to	C ctr ctr ctr ctr ctr ctr ctr ctr	11+4 Image: Control of the	Name C • cattle-cluster-agent-84c8857/07-ctgk2 C • cattle-cluster-agent-94c8858/07-trykk P • pdraftar-64d807392-27ht P • portext-settre-altrnmage-0 P • prometheuskube-state-metrics-17c68b8841-8976k P 0.0 (708207c50b/48b642de4802z818/7333954e90bc-7f511a09ff V v0.0 1708207c50b/48b642de4802z818/7333954e90bc-7f511a09ff V v0.0 1708207c50b/48b642de4802z818/7333954e90bc-7f511a09ff V v0.0 1708207c50b/48b642de4802z818/7333954e90bc-7f511a09ff V v0.0 1708207c50b/48b642de42de4802z818/7333954e90bc-7f511a09ff V v0.0 1708207c50b/48b642de42de4802z818/7333954e90bc-7f511a09ff V v0.0 1708207c50b/48b642de4802z818/7333954e90bc-7f511a09ff V v0.0 1708207c50b/48b642de4802z818/7333954e90bc-7f511a09ff V v0.0 1708207c50b/48b642de4802z818/7333954e90bc-7f511a09ff V v0.0 1708207c50b/48b642de4802z818/73335954e90bc-7f514005444738bc-7f56 <td></td> <td>Last* 2.2 2 MB 2.2 2 MB</td>		Last* 2.2 2 MB 2.2 2 MB

Desde el uso de nuestra CPU, RAM y el almacenamiento ocupado, hasta el uso de CPU y RAM de los pods, los procesos de CPU y RAM que se están ejecutando...

Con esto, podemos ver la integración de nuestro cluster de manera rápida con las métricas mediante Grafana y Prometheus de nuestro cluster desplegado mediante Rancher + K3s.





3.4.3. Desplieque y configuración del Runner en Gitlab

Ya con Gitlab y Runner instalados, pasamos a la creación del Registry privado donde almacenaremos las imágenes generadas por nuestro Runner cada vez que se dispare. Para ello, crearemos dicho Registry privado en la máquina manager como hemos comentado al principio de la documentación.

Primero, crearemos el entorno de trabajo antes de ponernos con la creación del escenario mediante docker compose. Los siguientes pasos que realizaremos, los haremos como root o superusuario.

```
$ mkdir ~/private-registry
$ mkdir ~/private-registry/registry-data
```

Accederemos al directorio del mismo y crearemos el fichero docker-compose.yaml donde crearemos el respectivo contenedor.

```
$ cd ~/private-registry
$ nano docker-compose.yml
version: '3'
services:
  registry:
      image: registry:latest
      ports:
      - "5000:5000"
      environment:
      REGISTRY_AUTH: htpasswd
      REGISTRY_AUTH_HTPASSWD_REALM: Registry
      REGISTRY_AUTH_HTPASSWD_PATH: /auth/registry.password
      REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY: /registry-data
      volumes:
      - ./auth:/auth
      - ./registry-data:/registry-data
```

Este archivo docker-compose.yaml configura un servicio de registro privado de contenedores Docker usando la imagen registry:latest. Además, expone el puerto 5000 del contenedor al host, permitiendo el acceso al registro a través de http://localhost:5000 (en este caso seria http://[ip_ext_máquina_manager]:5000).





Implementamos la autenticación básica usando un archivo de contraseñas ubicado en ./auth/registry.password y almacena las imágenes de contenedores en ./registry-data, proporcionando persistencia de datos.

Tras esto, desplegamos el escenario con el siguiente comando:

\$ docker compose up -d

Con esto, ya tendríamos nuestro Registry privado pero antes, deberemos cambiar la contraseña del usuario de este mismo ya que nos lo genera por defecto cuando levantamos el contenedor de Docker mediante el comando htpasswd.

```
<u>$ apt install apache2-utils (si no está instalado)</u>
$ htpasswd -Bc auth/registry.password admin
```

Nos pedirá una contraseña que tendremos que acordarnos para el registry que crearemos con Rancher posteriormente. Seguido de esto, pasamos a la máquina de gitlab, la cual tenemos el proyecto, y clonamos el repo en la máquina en local.

\$ git clone http://[ip_ext_máguina_manager]/root/mi-primera-app.git

Para poder clonarnos el proyecto, deberemos ingresar el usuario y la contraseña al acceder a gitlab. El usuario es **root** y la contraseña se puede obtener en el directorio /etc/gitlab/initial_root_password.

```
oot@gitlab:/home/pepe# git clone http://192.168.122.208/root/mi-primera-app.git
Clonando en 'mi-primera-app'...
Username for 'http://192.168.122.208': root
Password for 'http://root@192.168.122.208':
remote: Enumerating objects: 6, done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 6 (from 1)
Recibiendo objetos: 100% (6/6), listo.
 root@gitlab:/home/pepe#
```

Con esto, ya tendríamos clonado el proyecto. Con el repo copiado, añadimos los archivos en cuestión que son los siguientes:



- index.html: una plantilla de ejemplo para ver el funcionamiento de la integración continua.

```
$ nano index.html
<!DOCTYPE html>
<html lang="es">
<head>
     <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Página de Ejemplo</title>
      <style>
      body {
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
            height: 100vh;
            margin: 0;
            background-color: #f0f0f0;
            font-family: Arial, sans-serif;
      h1, h2 {
            color: #8c52ff;
            text-align: center;
            border: 2px solid #8c52ff;
            padding: 20px;
            border-radius: 10px;
            background-color: white;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
            margin: 10px;
      h2 {
            color: #21ef80;
            border: 2px solid #21ef80;
      </style>
</head>
<bodv>
      <h1>Prueba de funcionamiento CI/CD mediante Gitlab con Rancher +
K3S</h1>
      <h2>Realizado por Jose Carlos Rodríguez Cañas</h2>
</body>
</html>
```



- Dockerfile: donde copiaremos el fichero index.html que hemos modificado en un imagen por defecto de servidor web. En mi caso es una imagen de **nginx**.

```
$ nano Dockerfile
```

```
FROM nginx
COPY index.html /usr/share/nginx/html
```

- .gitlab-ci.yml: fichero que lee el runner para poder ejecutar el pipeline.

```
$ nano .gitlab-ci.yml
stages:
 - build-containers
 - deploy
variables:
 DOCKER_IMAGE: "192.168.122.3:5000/mi-primera-app:v${CI_PIPELINE_ID}"
 K8S_NAMESPACE: "default"
 K8S_DEPLOYMENT: "mi-primera-app"
docker_build:
 stage: build-containers
 image:
      name: gcr.io/kaniko-project/executor:debug
      entrypoint: [""]
 before_script:
      - cp "$KANIKO_SECRET" /kaniko/.docker/config.json
 script: >
      /kaniko/executor --context "${CI_PROJECT_DIR}" --dockerfile
"${CI_PROJECT_DIR}/Dockerfile" --destination "${DOCKER_IMAGE}"
deploy:
 stage: deploy
 image:
     name: bitnami/kubectl:latest
      entrypoint: [""]
 before_script:
      - cp "$KUBECONFIG_SECRET" /.kube/config
 script:
      - kubectl -n $K8S_NAMESPACE set image deployment/$K8S_DEPLOYMENT
$K8S_DEPLOYMENT=$DOCKER_IMAGE
```





Como podemos ver, el fichero .gitlab-ci.yml es el fichero más importante de nuestro, es por ello, que explicaré su debido funcionamiento.

Este archivo gitlab-ci.yml configura una pipeline de CI/CD en GitLab con dos etapas principales: construcción de contenedores y despliegue. A continuación, explicaré cada sección del archivo:

- Etapas:

- build-containers: Esta etapa es para la construcción de contenedores Docker.

- deploy: Esta etapa es para desplegar la aplicación en Kubernetes.

- Variables:

- DOCKER_IMAGE: Definimos la imagen Docker con la dirección del registro (192.168.122.3:5000) y la etiqueta basada en el ID del pipeline (v\${CI_PIPELINE_ID}).

K8S_NAMESPACE: Definimos el espacio de nombres en Kubernetes donde se desplegará la aplicación (default).

K8S_DEPLOYMENT: Definimos el nombre del despliegue en Kubernetes (mi-primera-app).

NOTA: Hay que comentar que hay 2 variables que no están declaradas en este pipeline que son KANIKO_SECRET y KUBECONFIG_SECRET. Cuando terminemos de subir los cambios y conoceré explicaré como declaramos estas variables generales.

- Job docker_build:

- Pertenece a la etapa build-containers.

- Usa la imagen gcr.io/kaniko-project/executor:debug para construir el contenedor.

- Antes de ejecutar el script, copia el secreto de Kaniko (\$KANIKO_SECRET) para la autenticación con el Docker registry.

- El script ejecuta Kaniko para construir la imagen Docker usando el Dockerfile en el directorio del proyecto y sube la imagen al registro definido.





- Job deploy:

- Pertenece a la etapa deploy.
- Usa la imagen **bitnami/kubectl**:latest para el despliegue.

- Antes de ejecutar el script, copia el secreto de configuración de kubectl (\$KUBECONFIG_SECRET) para la autenticación con Kubernetes.

- El script usa **kubectl** para actualizar la imagen del despliegue en Kubernetes con la nueva imagen construida.

Habiendo explicado el uso de estos ficheros y qué contenido tienen, pasamos a subir los cambios al proyecto.

\$ cd mi-primera-app \$ git add . \$ git commit -am "Contenido añadido" \$ git push origin main

Como no estamos logueados en la terminal para poder hacer el comando **git push**, tendremos que acceder a nuestra aplicación de gitlab y editaremos el perfil del administrador.



Tras esto, bajamos y copiamos el email general por nuestra aplicación y lo introducimos en este comando:





root@gitlab:/home/pepe/mi-primera-app# git configglobal user.email "gitlab_admin_b30248@example.
root@gitlab:/home/pepe/mi-primera-app# git commit -am "Contenido añadido"
[main 2ea2838] Contenido añadido
3 files changed, 69 insertions(+)
create mode 100644 .gitlab-ci.yml
create mode 100644 Dockerlle
create mode 100644 index.ntml
root@gtttab:/nome/pepe/mt-primera-app# gtt push origin Main
Username for 'http://192.168.122.208': Foot
Password for http://root@192.108.122.208 :
Enumerando objetos: 6, listo.
Containdo objetos: 100% (0/0), tisto.
Compression della usando nasta 4 nitos
Complemented objetos: 100% (4/4), ESEO.
$T_{0} = 5$ (dolta a) source of (dolta b) pack source of a
To http://192.169.122.209/root/mi-primera-app.oit
44419a82ea2838 main -> main



$\leftarrow \rightarrow \mathbf{C}$	🔿 👌 192.168.122.208/roo	ot/mi-primera-app		
🕣 Importar marcadores 🛭 😆 Cor	nenzar a usar Firefox			
⊌ □ + ○	Administrator / ml-primer	ra-app		
D :: C		M and and a second of		
Q Search or go to		mi-primera-app		25
Project		° main ∽ mi-primera-app / + ∽		History Find file Edit ~ Code ~
M mi-primera-app				
🖈 Pinned 🗸		Contenido añadido Administrator authored 5 minutes ago		(2) 2ea28382 [^{ey}
Issues		5		
Merge requests 0		Name	Last commit	Last update
යි Manage >		🦊 .gitlab-ci.yml	Contenido añadido	5 minutes ago
🛱 Plan >		🕒 Dockerile	Contenido añadido	5 minutes ago
> Code >	DEADINE and			1 day and
	README.Md	README.md	Opdate README.md	i day ago
		🥫 index.html	Contenido añadido	5 minutes ago
© Deploy >		README.md		
④ Monitor >		mi-primera-app		
는 Analyze >		frame and		
Ø Settings >		Primer acercamiento a CI/CD) con Gitlab montado en el clus	ster local con Rancher y K3s

Antes de añadir las variables globales de Gitlab, tendremos que añadir a todas las máquinas que tenemos instalado Docker el siguiente demonio para que acceda el registry de manera no segura por el protocolo HTTP.

Para que los cambios surjan, tiene que reiniciarse el servicio y volver a levantar el compose del registry.

```
$ nano /etc/docker/daemon.json
  "insecure-registries": ["[ip_externa_manager]:5000"]
$ sudo systemctl restart docker
```





Pasamos a la configuración de las variables generales que nos faltan para que funcione nuestro pipeline. Para ello, accederemos Settings -> CI/CD y al apartado de Variables -> Expand.

$\leftarrow \rightarrow C$	0	2 192.168.122.208/root/mi-primera-app/-/settings/ci_cd	▣ 蒅 ☆
-🗈 Importar marcadores.	👈 Come	enzar a usar Firefox	
₩ □	+ ()	Administrator / ml-primera-app / CI/CD Settings	
D 11	G	Runners	Expand
Q Search or go to	D	Runners are processes that pick up and execute CI/CD jobs for GitLab. What is GitLab Runner?	Expand
Merge requests	0	Artifacts	Expand
路 Manage 岗 Plan	>	A job artifact is an archive of files and directories saved by a job when it finishes.	
Code	>	Variables	Collapse
🥝 Build	>	Variables store information that you can use in job scripts. Each project can define a maximum of 8000 variables. Learn more. Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature can help reduce the risk of accidentally	y exposing
⊕ Secure ⊡ Deploy	>	variable values, but is not a guaranteed method to prevent malicious users from accessing variables. How can I make my variables more secure? Variables can have several attributes, Learn more.	
Operate	>	Protected: Only exposed to protected branches or protected tags.	
Monitor Analyze	>	 masked: Holden in job logs. Must match masking requirements. Expanded: Variables with \$ will be treated as the start of a reference to another variable. 	
Settings		CI/CD Variables ↔ 0 Add v	variable
General		Key ↑ Value Environments	Actions
Integrations Webhooks		There are no variables yet.	

Las variables que añadiremos son KANIKO_SECRET y KUBECONFIG_SECRET que ambas son de tipo File y se encuentran en los siguientes directorios.

- KANIKO_SECRET: se encuentra en el fichero /root/.docker/config.json en la máquina manager, el cual sí no existe lo que tienes que hacer es loguearte a tu registry mediante ip con este comando:

\$ docker login [ip_externa_manager]:5000

Cuando te logues, este fichero se creará automáticamente y es por ello que tendrá el siguiente contenido:





Este contenido lo copiamos y lo añadimos a la variable de gitlab.

$\leftarrow \rightarrow \mathbf{G}$	0	192.168.122.208/root/mi-primera-app/-/settings/ci_cd			🗉 💫 ☆
- Importar marca	dores 📦 Come	nzar a usar Firefox			
₩	□ + ○	Administrator / mi-primera-app / CI/CD Settings			
D 11		Variables			(
Q Search o	r go to	Variables store information that you can use in ju	ob scripts. Each project can define a maximum o	f 8000 variables. Learn more.	
Issues		Variables can be accidentally exposed in a job lo variable values, but is not a guaranteed method	og, or maliciously sent to a third party server. Th to prevent malicious users from accessing varia	e masked variable feature can help reduce the ris bles. How can I make my variables more secure?	sk of accidentally exposing
weige reques	0	Variables can have several attributes. Learn more	re.		
<u>රි</u> සි Manage	>	Protected: Only exposed to protected b	ranches or protected tags.		
🛱 Plan	>	Masked: Hidden in job logs. Must match i	masking requirements.		
> Code	>	Expanded: Variables with \$ will be treated	ed as the start of a reference to another variable		
🥝 Build	>				
① Secure	>	CI/CD Variables <> 1		Reveal	values Add variable
Deploy	>	Кеу 🛧	Value	Environments	Actions
Operate	>	KANIKO_SECRET	***** [⁰ 1	All (default) 🔓	
🖽 Monitor	>	(File) (Protected) (Expanded)			
h Analyze	>				

- KUBECONFIG_SECRET: este fichero se ubica en la interfaz de Rancher, en el siguiente icono que pone Download Kubeconfig.

\rightarrow	с	O 🗛 htt	:ps://192.168.122.3:	:8443/dashboard/c/c-m-4hqm267j/e	xplorer#cluster-events						☆
mportar marcadores 🐞 Comenzar a usar Firefox											
E	<u> c</u> luster-k3s							Only User Namespaces		Ţ	۷ 🗈 ն
	Cluster										Download KubeConfig
<u> </u>	Projects/Namespaces		Cluster Da	ashboard							
3 1	Nodes Cluster and Project M	1 1embers	Provider: K3s	Kubernetes Version: v1.29.5 +k3s1	Created: 1.2 days ago						Install Monitoring

Tendremos que descargarlos en nuestra máquina local y añadir todo el contenido en crudo en la variable que crearemos en Gitlab.

← → C O	2 192.168.122.208/root/mi-primera-app/-/settings/ci_cd			目本公	⋓							
🕣 Importar marcadores 🐞 Come	nzar a usar Firefox											
⊌ □ + ○	Administrator / mi-primera-app / CI/CD Settings											
D 11 E	Variables				Collapse							
Q Search or go to	Variables store information that you can use in job scripts. Each pr	oject can define a maximum of 8000 variab	es. Learn more.									
Merge requests	Variables can be accidentally exposed in a job log, or maliciously s variable values, but is not a guaranteed method to prevent malicion	ent to a third party server. The masked vari us users from accessing variables. How car	able feature can help reduce the risk of acc I make my variables more secure?	identally exposing								
රී Manage >	Variables can have several attributes. Learn more.											
🗇 Plan >	Protected: Only exposed to protected branches or protect Masked: Hidden in job logs. Must match masking requireme	Protected: Only exposed to protected branches or protected tags. Masked: Hidden in job logs. Must match masking requirements.										
Code >	Expanded: Variables with \$ will be treated as the start of a	reference to another variable.										
			Development	Add controls								
♥ Secure >	CI/CD Variables 422		Reveal values	Add variable								
© Deploy >	Key ↑	Value	Environments	Actions								
Operate >	KANIKO SECRET	***** 🛱	All (default)	2 🕅								
Monitor >	File Protected Expanded	*LJ	, m (assess) =[]									
Analyze >	KUBECONFIG_SECRET	***** ß	All (default) l_{Ω}^{n}	0								

Con esto ya tendríamos la configuración de Gitlab y ru respectivo Runner de manera correcta.





4. Demostración final

En esta sección, llevaremos a cabo una demostración práctica que abarca varios aspectos clave de la integración y monitorización de aplicaciones. Es decir, vamos a realizar una **prueba de la integración continua con Gitlab mediante una aplicación con un clúster en Rancher y monitorizado con Grafana y Prometheus**.

Pero antes de realizar la integración, vamos a configurar el deployment para poder correr la aplicación y secret para poder acceder al **registry** privado.

Para ello, primero, tendremos que acceder a la máquina **master** donde tenemos ubicado el cluster de **K3s**, modificamos y añadimos el siguiente archivo:

<pre>\$ nano /etc/rancher/k3s/registries.yaml</pre>
mirrors:
"http://[ip_externa_manager]:5000":
endpoint:
<pre>- "http://[ip_externa_manager]:5000"</pre>

Tras esto, reiniciamos la máquina master para que actualicen los cambios correctamente.

Una vez terminada, pasamos a crear el **secret** para el **registry** privado. Para ello accederemos al apartado de **Storage -> Secrets** y dándole a crear.

$\leftarrow \ \ \rightarrow$	c C	http	s:// 192.168.122.3 :8443/dashboard/c/c-m-4hqm267j/explorer/secret				☆			⊚ :	Ŀ	ി ⊒
-D Impor	tar marcadores 👈 Come	enzar a usa	Firefox									
≡	<u> </u> cluster-k3s				Only User Namespaces	1	٤	ľ		ρ c	n	: 😂
	Cluster										_	
"	Workloads		Secrets 🌣									
СКЗ	Apps						_					
	Service Discovery		∠ Download YAML				=	•	F			
	Storage											
	PersistentVolumes		☐ State	Kind 🗘	Data					Age		
	StorageClasses											
	ConfigMaps		Namespace: default									
	PersistentVolumeClaims			Opaque	admin-password, admin-user, Idap-toml					1.3	days	
	Secrets	(1) 3		Helm Release	(none)					1.3	davs	
	Policy											





Tras esto, crearemos un Registry y lo seleccionamos.

$\leftarrow \ \ \rightarrow$	c O	🔏 http	s:// 192.168.122.3	:8443/dashboard/c/c-m-4hqm267j/explorer/secret/create					☆		⊚	¥ @) එ ≡
- Impor	tar marcadores 👈 Comen	zar a usa	r Firefox										
≡	<u> c</u> luster-k3s						Only User Namespaces Y			Ö	Q	m	: 😂
п скз	Cluster Workloads Apps		Secret: Cr	eate									
۳	Service Discovery Storage PersistentVolumes StorageClasses	> 2 1	C	Custom Type Create a Secret with a custom Type Opaque		HTTP	HTTP Basic Auth Authentication with a username and pa						
	PersistentVolumeClaims Secrets	(H) 2 (H) 3	0			R		iner imag	ges				
	Policy More Resources	>	SSH	SSH Key Public key and private key for SSH authentication	More Info 团	TLS	TLS Certificate Store a certificate and key for TLS						

Por último, para crearlo tendremos que indicar la ip de la máquina manager con el puerto 5000 de nuestro **registry** más el usuario y la contraseña que cambiamos mediante el comando htpasswd.

$\leftarrow \rightarrow$	c O	<u>6</u> ⊶ ∣	https:// 192.168.122.3 :8443/da	ashboard/c/c-m-4hqm267j,	/explorer/secret/create#data				厽		0	¥ ,	⊜ ഫ ≡
-9 Impor	tar marcadores 👈 Comen	zar a usa	r Firefox										
≡	Luster-k3s						Only User Namespaces		٤ ا) (D	Q	m	÷ 😫
СКЗ	Cluster Workloads Apps		Secret: Create Reg	gistry	Name*	Descri	ation						
	Service Discovery Storage PersistentVolumes StorageClasses	> ~ 2	default		reg-docker		xx you want that better describes this reso						
	ConfigMaps PersistentVolumeClaims Secrets	(H) 5 (H) 2 (H) 3	Data Labels & Annotations	Data Custom DockerHub									
	Policy More Resources	>		Quayið Artifactory Registry Domain Name * 192.168.122.3:5000 Username admin			Passoord						

Pasamos a crear el deployment desde la interfaz accediendo Workloads -> Deployments y dándole a crear.

$\leftarrow \rightarrow$	С	O 🔒 https	s:// 192.168.122.3 :84	443/dashboard/c/c-m-4hqm2	67j/explorer/	/apps.deployment						☆		⊚	⊻	ء ھ	<u>}</u> ≡
- Impoi	rtar marcadores 👈	Comenzar a usar	r Firefox														
≡	📙 cluster-k3s						Only User	Namespace	s			٤ [Q	m		2
	Cluster																
n	Workloads		Deploymen	ts ☆												Cre	
СКЗ	CronJobs																
	DaemonSets		O Redeploy									=	•				
	Deployments	(e) 4															
	Jobs	(m) O	State 🗘	Name 🗘		Image 🗘	Rea	dy⊖ Up T	o Date 🗘	Available 🗘	Restarts	s Age≎		Health			
	StatefulSets																
	Pods		Namespace: defau	lt													
	Apps	>				grafana/grafana:11.0.0						1.3 da		_		••	÷
	Service Discovery					registry.k8s.io/kube-state-metrics/kube-state-metrics:v2.12.0	1/1				6	1.3 da	vs	_			:
	Storage																_





Por último, en la configuración del deployment **añadiendo nombre de la imagen** que queremos importar, el **registry** que hemos creado anteriormente y el puerto del contenedor el 80 (por defecto por nginx) al **Nodeport** 30598 (se puede autogenerar si no indicamos alguno).

<u> c</u> luster-k3s						0	only User Namespa	ces 🗸	r 1	٤	ĥ	Û	ם م	n :	3
Cluster															
Workloads		Deployment: Creat	e												
CronJobs															
DaemonSets		Namespace*		Name*											
Deployments	(•) 4														
Jobs															
StatefulSets			0												
Pods															
Apps			hi-primera-app + Add												
Service Discovery		General	<u> </u>												
Storage		Health Check	General												
Policy															
More Resources			mi-primera-app				Standard Co								
			Image												
							Pull Policy								
			192.168.122.3:5000/r	mi-primera-app:v2			Always								
			Pull Secrets												
			Networking	es the container or define a new f	national pamod past of that I	humana will be	out the end	uithin the container.		te eur					
			If Cluster IP, LoadBalano	er, or NodePort is selected, a Servi	ce is automatically created th	at will select th	e Pods in this work	load using labels.	sexpected	nto run.					
			Service Type								otocol				
			Node Port		my-port			80	0	тс	P		l0598 🗘	R	

Creamos el deployment y ya tendríamos todos los cambios de manera que nos faltaría hacer la integración continua. Para ello, realizaremos un **commit** añadiendo un punto al primer h1 de nuestro **index.html** y subiremos el cambio.



Cuando realicemos el push, automáticamente nuestro Runner disparará los **jobs** correspondientes y los ejecutará de manera correcta.



$\leftarrow \rightarrow \mathbf{G}$	192.168.122.208/root/mi-primera-app/-/pipelines/12
🕣 Importar marcadores 🖕 Come	nzar a usar Firefox
⊌	Administrator / mi-primera-app / Pipelines / #12
D II R	Contenido añadido, versión 1
Q Search or go to	🖉 Passed Administrator created pipeline for commit c832ce49 🗛 2 minutes ago, finished 1 minute ago
Project	For main
M mi-primera-app	latest CO 2 jobs 🐧 18 seconds, queued for 1 seconds
🖈 Pinned 🗸 🗸	Pineline Needs Jobs 2 Tests 0
Issues	
Merge requests 0	
සී Manage >	build-containers
ë Plan →	O docker_build C O deploy C
Code >	
🤣 Build 🗸	
Pipelines	
lobs	

Como vemos se ha disparado correctamente y si accedemos a la URL siguiente podremos ver la aplicación funcionando de manera correcta.

<pre>\$ http://[ip_externa_master]:[puerto_node_port]/</pre>		
En mi caso		
\$ http://192.168.122.31:30598/		
← → C O A 192.168.122.31:30598	슙	© ¥
ා gontar marcadores 🧆 Comenzar a usar Firefox		
Prueba de funcionamiento CI/CD mediante Gitlab con Rancher + K3S.		
Realizado por Jose Carlos Rodríguez Cañas		

Como podemos ver, se ha desplegado de manera correcta pero para verlo de manera más explícita, vamos a ver un cambio en la aplicación mediante el index.html





Le vamos a añadir al h1 un v2 al final para ver que se despliega de manera automática con el Runner.

root@git	tlab:/home/pepe/mi-primera-app# nano index.html
root@git	tlab:/home/pepe/mi-primera-app# cat index.html
DOCTYF</td <td>PE html></td>	PE html>
<html la<="" td=""><td>ang="es"></td></html>	ang="es">
<head></head>	
	<meta charset="utf-8"/> <meta content="width=device-width, initial-scale=1.0" name="viewport"/> <title>Página de Ejemplo</title> <style></td></tr><tr><td></td><td>body {</td></tr><tr><td></td><td>display: flex;</td></tr><tr><td></td><td>justify-content: center;</td></tr><tr><td></td><td>align-items: center;</td></tr><tr><td></td><td>flex-direction: column;</td></tr><tr><td></td><td>height: 100vh;</td></tr><tr><td></td><td>margin: 0;</td></tr><tr><td></td><td>Dackground-color: #tururu;</td></tr><tr><td></td><td>ront-ramity: Artal, sans-serir;</td></tr><tr><td></td><td>۲ h1 h2 ۲</td></tr><tr><td></td><td>11, 12</td></tr><tr><td></td><td>text-alion: center:</td></tr><tr><td></td><td>border: 2px solid #8c52ff:</td></tr><tr><td></td><td>padding: 20px:</td></tr><tr><td></td><td>border-radius: 10px;</td></tr><tr><td></td><td>background-color: white;</td></tr><tr><td></td><td>box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);</td></tr><tr><td></td><td>margin: 10px;</td></tr><tr><td></td><td>}</td></tr><tr><td></td><td>h2 {</td></tr><tr><td></td><td>color: #21ef80;</td></tr><tr><td></td><td>border: 2px solid #21ef80;</td></tr><tr><td></td><td>} </td></tr><tr><td></heads</td><td></style>
loodys	<pre><h1>Prueba de funcionamiento CI/CD mediante Gitlab con Bancher + K3S v2</h1></pre>
	<h2>Realizado por Jose Carlos Rodríguez Cañas</h2>
root@git	tlab:/home/pepe/mi-primera-app# git add .
root@git [main 38	tlab:/home/pepe/mi-primera-app# git commit -am "Contenido añadido, versión 2" 345b22] Contenido añadido, versión 2
1 Tile	changed, 1 insertion(+), 1 deletion(-)
llsername	for 'http://192 168 122 208'; coot
Dassword	f = 101 + 101 + 102 + 103 + 122 + 208 + 1001
Fnumerar	ado objetos: 5 listo.
Contando	o objetos: 100% (5/5), listo.
Compresi	ión delta usando hasta 4 hilos
Comprimi	lendo objetos: 100% (3/3), listo.
Escribie	endo objetos: 100% (3/3), 316 bytes 316.00 KiB/s, listo.
Total 3	(delta 2), reusados 0 (delta 0), pack-reusados 0
To http:	://192.168.122.208/root/mi-primera-app.git
c0320	ce43845b22 main -> main
root@git	tlab:/home/pepe/mi-primera-app#

Tras éllo, saltará dicho Runner y se actualizará la aplicación de manera rápida y sencilla.



$\leftarrow \rightarrow \mathbf{G}$	2 🔁 192.168.122.208/root/mi-primera-app/-/pipelines/13
🕣 Importar marcadores 🖕 Com	enzar a usar Firefox
⊌ □ + ○	Administrator / mi-primera-app / Pipelines / #13
D 11 2	Contenido añadido, versión 2
Q Search or go to	🕑 Passed Administrator created pipeline for commit 3845b224 🖞 1 minute ago, finished 1 minute ago
Project	For main
M mi-primera-app	latest CO 2 jobs 🐧 16 seconds, queued for 0 seconds
🖈 Pinned 🗸	Pipeline Needs Jobs 2 Tests 0
lssues	
Merge requests 0	
å Manage →	build-containers
ট Plan >	Image: Constraint of the second se

Volvemos a acceder a la aplicación y ...

$\leftarrow o \mathbf{C}$ -D Importar marcadores	○ 2 192.168.122.31:30598 Comenzar a usar Firefox	☆
	Prueba de funcionamiento CI/CD mediante Gitlab con Rancher + K3S v2	
	Realizado por Jose Carlos Rodríguez Cañas	

Con ésto, ya tendríamos la integración continua mediante Gitlab en nuestro cluster de alta disponibilidad de K3s orquestado por Rancher.

Si queréis ver el video de como lo hago en tiempo real, accede a este video creado por mi para comprobar la veracidad del despliegue:

\$ https://voutu.be/gz273b_rsNw





5. Dificultades encontradas

Con respecto a las dificultades encontradas, realmente han sido cuestión de tiempo y de plantear opciones a la hora de montar nuestro cluster de alta disponibilidad.

Algunos de los ejemplos de conflictos o dificultades encontradas pueden ser la conexión del Registry privado con nuestro Gitlab, la integración continua con nuestro cluster de manera automática...

Éstas y muchas más pueden ser las dificultades que me he podido encontrar en mi proyecto integrado, pero realmente los problemas que me han generado mayor incertidumbre son los siguientes.

5.1. Conexión del cluster con Rancher

Una de las dificultades que me han dado tantísimos problemas ha sido la instalación del cluster con nuestro Rancher.

Como ya comentamos en la instalación del mismo, lo haríamos con **Docker** y ese fue el problema al principio. Lo intenté desplegar de la manera más básica como es la siguiente.

```
$ docker run -d --restart=unless-stopped \
    -p 80:80 -p 443:443 \
    --privileged \
    rancher/rancher:stable
```

Lo que yo no contaba es que este contenedor Docker suele estar configurado para poder utilizar **HTTPS** aunque no disponga de certificado.

Por ello que cuando instalamos el curl de importación del mismo cluster de K3s, nos dará un problema **curl 60** por la URL **HTTPS** que utiliza.

- Info del error en cuestión

La máquina master no encuentra ningún certificado firmado por la **CA** ya que ese certificado es autofirmado en la máquina manager.





Una de las soluciones que apliqué para el correcto funcionamiento de Rancher fue **añadir mi certificado y mi clave de mi dominio comprado en IONOS** (pepepfoter15.es) ya que está firmado por una CA que puede reconocer todas las máquinas del cluster mediante la salida al exterior.

Ésto lo haría copiando los certificados de mi máquina local a la máquina manager y generar un **volumen bind mount** de los mismos certificados.

```
$ sudo docker run -d --name rancher --restart=unless-stopped \
    -p 8080:80 -p 8443:443 \
    -v
/etc/ssl/rancher/pepepfoter15.es_ssl_certificate.cer:/etc/rancher/ssl/pepepf
oter15.es_ssl_certificate.cer \
    -v
/etc/ssl/rancher/_.pepepfoter15.es_private_key.key:/etc/rancher/ssl/_.pepepf
oter15.es_private_key.key \
    -v /var/lib/docker/rancher:/var/lib/rancher \
    --privileged \
    rancher/rancher:stable
```

Tras éllo, el problema se vió resuelto en instantes.

5.2. Configuración de Gitlab y su respectivo Runner

A la hora de la configuración de nuestro Gitlab, tenía una primera opción de instalación, realizada mediante el chart de Helm como realizamos con los contenedores de Grafana y Prometheus.

Uno de los inconvenientes de este mismo era el proceso de instalación ya que al instalarlo mediante el protocolo **HTTPS** y crear un **ingress** en nuestro **Rancher**, nos sucedía un problema con las URL a la hora de cargar la hora de estilos.

- Info del error en cuestión

Como podemos ver en la siguiente imagen el mismo **ingress**, mostraba el contenido de los contenedores pero sin ningún tipo de hoja de estilos, haciendo la instalación de este mismo inútil.



\$ El recurso de

"https://gitlab.pepepfoter15.es/assets/webpack/runtime.bad33c1a.bundle.js" fue bloqueado debido a una discordancia del tipo MIME ("text/html") (X-Content-Type-Options: nosniff).

Projects · GitLab	× +		~	_ @ x
$\leftrightarrow \rightarrow C$	A https://gitlab.pepepfoter15.es	\$4 \$2		⊛ ഹ =
Check your sign-up	restrictions			
Your GitLah instance allows a	wone to register for an account, which is a security risk on nublic facing GitLab instances. You should deactivate new sign uns if nublic users aren'	expected to register for an account		
Deactivate Acknowledge	······································			
https://gitlab.pepepfoter15.es/asse	ts/webpack/super_sidebar.d2a1eb80.chunk.js			
Grinspector ► Consola D D	epurador ᡝ Red () Editor de estilos (2) Rendimiento 🅕 Memoria 💾 Almacenamiento 🔺 Accesibilidad 🗱 Aplicación		ee vi	15 [] ••• X
A Ha fallado la carga del <script></script>				

El comando de instalación en cuestión era el siguiente:

<pre>\$ helm upgradeinstall gitlab gitlab/gitlab \ namespace gitlab \</pre>		
kubeconfig /etc/rancher/k3s/k3s.yaml \		
timeout 600s \		
set global.hosts.domain=pepepfoter15.es \		
set global.hosts.externalIP=[ip_externa_master] \		
set certmanager-issuer.email=pepepfoter15@gmail.com \		
set global.hosts.https=true \		
set global.ingress.enabled=true \		
set global.ingress.configureCertmanager=false \		
set certmanager.install=false \		
set global.gitlabVersion=17.0.1 \		
set nginx-ingress.enabled=false \		
set global.edition=ce \		

Para poder solucionar dicho problema, simplemente segmenté los servicios. Instalé una máquina Gitlab que tuviese dicha aplicación con su Runner y eliminé la instalación del chart de Helm del mismo.

Dicho proceso de instalación se encuentra en el apartado 3.3.4.





6. Conclusión

En este proyecto, se intenta buscar la construcción de una plataforma integral para gestionar clústeres de contenedores utilizando **Rancher y K3s**.

La combinación de estas herramientas nos permite implementar, monitorear y administrar aplicaciones de manera eficiente y segura, optimizar recursos y garantizar la máxima disponibilidad.

Elegir **Rancher** como orquestador y K3s como distribución de Kubernetes es clave para lograr una arquitectura robusta, escalable y fácil de administrar, especialmente en entornos con recursos limitados.



Durante el proyecto se han implementado soluciones de monitorización y visualización utilizando **Grafana y Prometheus**.

Además, la incorporación de **GitLab y GitLab Runner** para integración y entrega continuas (CI/CD), me ha simplificado enormemente el flujo de trabajo del desarrollador. **Esta automatización permite el desarrollo, las pruebas y la implementación de aplicaciones**.





La capacidad de **GitLab para gestionar todo el ciclo de vida del** desarrollo de software proporciona una visión unificada y flexible del proceso, desde la planificación hasta el lanzamiento.



Como resultado, el proyecto que he creado nos da una p**lataforma** sólida y eficiente para la gestión de clusters de contenedores, incorporando herramientas líderes en el mercado para garantizar disponibilidad, eficiencia y seguridad.

La implementación de **Rancher, K3s, Grafana, Prometheus y GitLab** ha creado un entorno unificado que respalda a los administradores y desarrolladores de sistemas y promueve prácticas de **DevOps** que optimizan el desarrollo y la implementación de aplicaciones.

Esta base no solo maneja las operaciones diarias, sino que también proporciona una base sólida para la expansión y la automatización de tareas.





7. Bibliografía

¿Qué es Kubernetes? ¿Qué es K3s? y Documentación de K3s

<u>¿Qué es Rancher?</u> <u>Página oficial - Rancher</u>

<u>Prometheus Exporters personalizados</u> <u>Curso de OpenWebinars - Grafana y Prometheus</u>

<u>Saber todo sobre el repositorio Git para DevOps - GitLab</u> ¿Qué es GitLab Runner?

Recomendaciones de arquitectura - Rancher Instalación del clúster de K3s con Rancher Instalación de Gitlab y Gitlab Runner Instalación de Grafana y Prometheus con Helm