

**Grado en Ingeniería en Sistemas de Información**

**TFG**

**Despliegue de Kubernetes con Ansible en entornos offline**

**Autor:** Sergio Picazo Serrano

**Tutor:** Óscar García Población

**Cotutor:** David de la Hoz Alías

2023

1. **Introducción y objetivos**
   1. **Introducción**

En este Trabajo de Fin de Grado se va a montar un clúster de Kubernetes mediante automatización con Ansible en un entorno offline montado On Premise.

A diferencia de las máquinas virtuales, los contenedores permiten desplegar, arrancar y parar aplicaciones más rápido, aprovechando mejor los recursos de hardware.

La motivación de este proyecto viene por el papel que están teniendo últimamente las tecnologías Cloud Native, en especial Kubernetes debido a que ayuda a las empresas a crear, escalar y administrar aplicaciones en la nube y mantener sus ciclos de vida dinámicos.

Según un artículo de Dominio de las Ciencias [1], se demuestra como actualmente, la manera más eficiente para el desarrollo y puesta en producción de aplicaciones es la implementación de microservicios contenerizados, orquestados en Kubernetes, y ya no los procesos tradicionales a través de monolitos.

Kubernetes permite el desarrollo y puesta en producción de aplicaciones de manera más rápida y con propiedades de escalamientos y alta disponibilidad.

La conclusión es evidente, desde su nacimiento está ganando una adopción masiva en todas las industrias, ayudando a las empresas a ofrecer soluciones de software con menos énfasis en la infraestructura.

Uno de los casos más destacados es el de Mercedes-Benz [2], hace años, los ingenieros de software se enfrentaban a tiempos difíciles en Mercedes-Benz: operaciones con hojas de cálculo, procesos manuales, infraestructuras crecidas y una gobernanza estricta. Una iniciativa popular de los ingenieros propuso el reto de cambiar las reglas del juego, y su bala de plata fue Kubernetes. Comenzaron con Kubernetes 0.9 en servidores gestionados hasta el día de hoy, donde manejan una plataforma on-premise self-service con cerca de 1000 clústeres en Cluster API. Apostaron por transformar un centro de datos con un equipo joven que, en su mayoría, desconocía los procesos empresariales, pero a través de una mezcla de visiones ingenuas y una fuerte creencia en el código abierto con mucha resistencia hizo que el proyecto fuera un éxito.

La clave de este trabajo reside en la necesidad de conseguir, de la manera más eficiente posible, aprovechando al máximo unos recursos limitados, desplegar un entorno en el que el mantenimiento de este y la gestión de aplicaciones sea lo más sencilla posible.

Esto se debe a que antiguamente la solución era mantener tus servicios y aplicaciones en un solo ordenador físico, llevando a obtener muchos problemas a las empresas.

Los contenedores nacieron para simplificar este proceso y ofreciendo virtualización ligera, generan el entorno mínimo necesario para aprovechar en mayor parte los recursos de la máquina física donde se ejecuta.

* 1. **Motivación**
  2. **Objetivo**

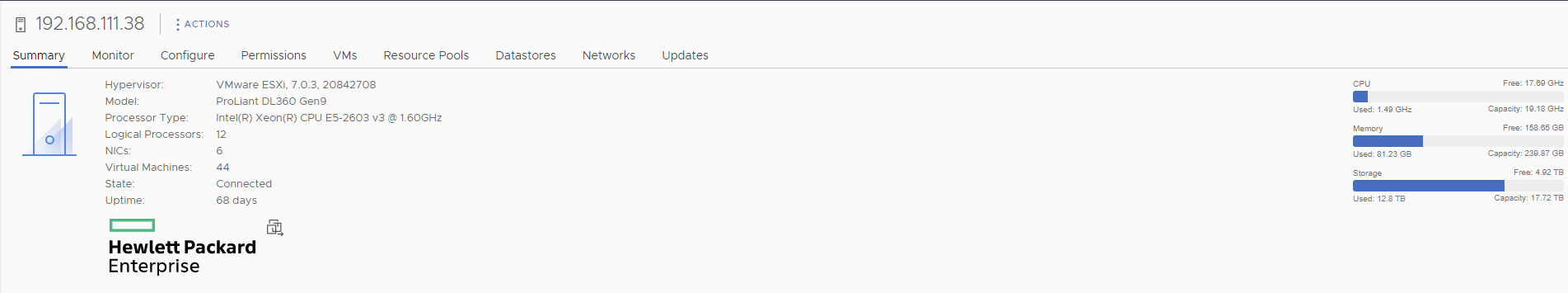
Este proyecto tiene como objetivo el diseño y construcción de una instalación automatizada de un clúster de Kubernetes, con aprovisionamiento automático, en un entorno de alta seguridad, en el que la conectividad a Internet es limitada o nula.

El objetivo al finalizar el proyecto es ser capaz de ofrecer un entorno totalmente funcional de laboratorio en el que se puedan hacer pruebas que conlleven riesgos y donde se puedan probar todo tipo de instalaciones de aplicativos para seguir aprendiendo sobre dicha plataforma, por tanto, un entorno de laboratorio es su principal campo de aplicación.

* 1. **Estructura del documento**

1. **Antecedentes**
2. Estudio acerca del uso de herramientas:
   1. Virtualización
   2. Contenedores
   3. Automatización
   4. IAC (infraestructura como código).
3. Identificar y dar a conocer las ventajas de los contenedores sobre otras tecnologías y sistemas de orquestación de contenedores, en este caso Kubernetes [3].
   1. **Contenedores**
   2. **Kubernetes**
   3. **Ansible**
4. **Desarrollo**
   1. **Prerrequisitos y preparación del entorno**

Información del equipo sobre el cual se desplegarán las VM (192.168.111.38):



* 1. **Planificación de uso de recursos (Requisitos de hardware y software)**
     1. **Creación de máquinas independientes (setup-host.sh)**

Se utilizarán máquinas RHEL (Red Hat Enterprise Linux) de laboratorio:

Máquina física: 192.168.111.38

Máquinas virtuales para el clúster:

| **IP** | **Hostname** | **Rol** |
| --- | --- | --- |
| 192.168.112.183 | master1.cin | Kubernetes Master y ETCD |
| 192.168.112.184 | worker1.cin | Kubernetes Worker1 |
| 192.168.112.185 | worker2.cin | Kubernetes Worker2 |
| 192.168.112.186 | registry.cin | Kubernetes Registry |
| 192.168.112.187 | loadbalancer1.cin | Kubernetes Load Balancer1 |
| 192.168.112.188 | loadbalancer2.cin | Kubernetes Load Balancer2 |
| 192.168.112.189 | vipa.cin | Kubernetes Ingress VIP (solo IP virtual, no es una máquina) |

Los datos de red correspondientes al laboratorio escogido son los siguientes. Cabe destacar que la instalación se debe hacer sin conexión a internet, proporcionando los paquetes y librerías necesarias.

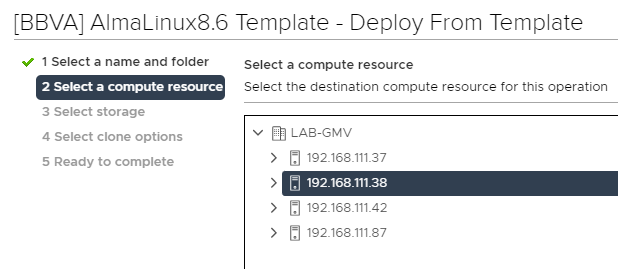
* Red: 192.168.112.0/24
* GW: 192.168.112.254
* DNS: 192.168.131.11
* NTP: 192.168.131.11
* Proxy Navegación: 192.168.131.13:80 | **10.200.140.230:80**

El template que utilicemos cómo imagen de RHEL 8.6 (AlmaLinux) tiene configurado como repositorio de RedHat uno que se ha instalado en otro equipo y que tiene los paquetes básicos.

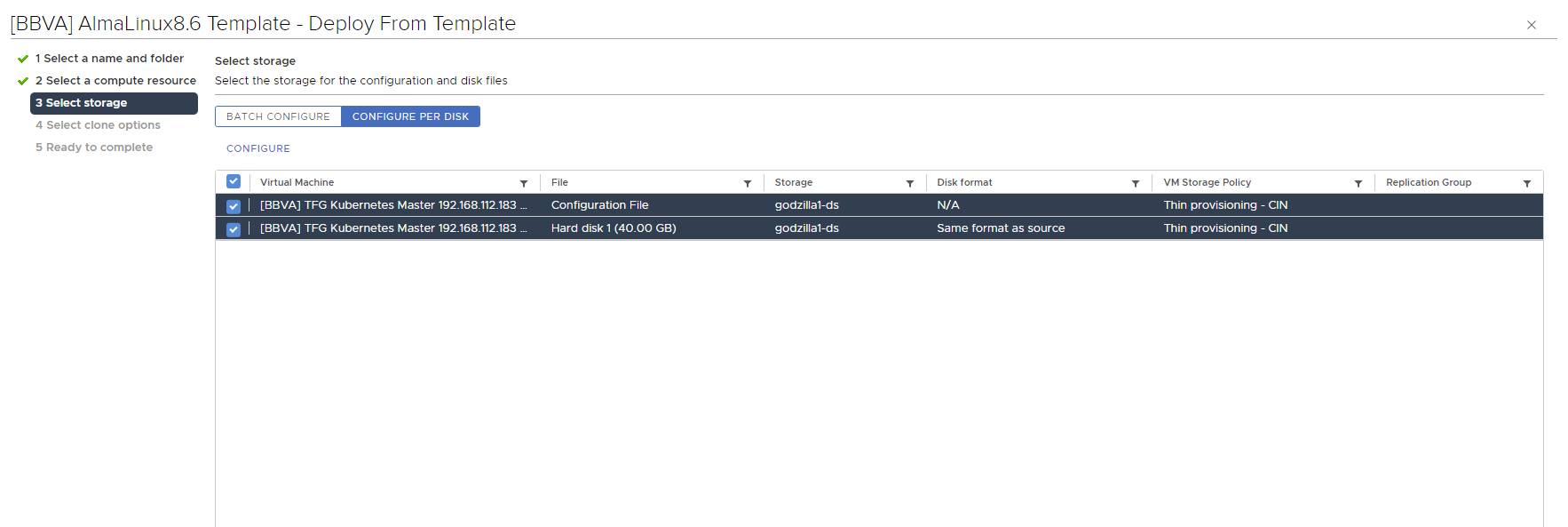
Procedemos a crear la VM del master1 desde la plantilla existente en el laboratorio.

Podemos crearla desde cero o a partir de la plantilla que ya existe llamada *“AlmaLinux 8.6 Template”*, que ya tiene el SO y el repositorio configurado.

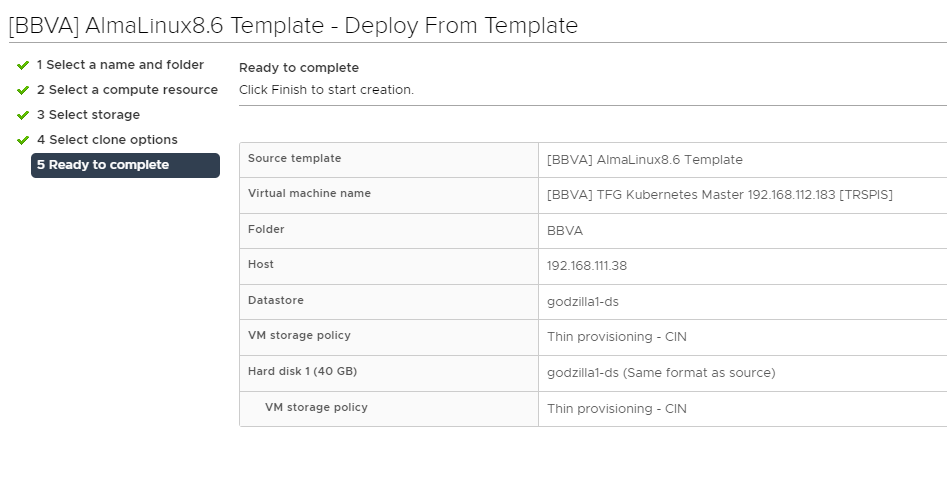
* Para ello pinchamos en el template y marcamos la opción *“New VM from This Template”*
* Seleccionamos el equipo sobre el que queremos desplegarla (192.168.111.38).



* Seleccionamos el almacenamiento



* Actualmente la máquina que usamos como plantilla tiene 4 CPUs, 16GB de RAM y un disco de 56GB.



Paso 1: Configuración inicial de las máquinas:

● Descripción: se configura la IP, hostname, usuario, grupo y directorio de trabajo para todas las máquinas.

● Configuración del archivo .bashrc de nuestro usuario, estableciendo el prompt y otros detalles mínimos.

● Creación del archivo .profile en el home de nuestro usuario, indicando que se cargue la configuración del .bashrc con cada login.

● Cambio del layout del teclado a español (originalmente está en teclado inglés americano).

● Añadir el grupo al archivo /etc/sudoers para que los usuarios que pertenezcan a él puedan ejecutar comandos como root con sudo. Si vamos a tener binarios que necesiten ejecutarse con sudo y que se encuentren en rutas que no estén en el PATH de sudo (comprobar con printenv), es necesario comentar la línea de secure\_path en el /etc/sudoers.

● Configuración de IP y hostname de las máquinas (explicado en Laboratorio GMV - Documentos de Google) + reinicio del servicio network. Importante mantener el UUID de la interfaz de red.

● Paquetes necesarios: ninguno.

● Scripts: [setup-host.sh]

Es necesario modificar el fichero **/etc/yum.repos.d/almalinux.repo** para añadir el proxy <http://10.200.140.230:80> y así salir a internet mediante él.





vi /etc/sysconfig/network-scripts/ifcfg-ens192

Y reiniciamos el servicio de red

*systemctl restart network.service*

*reboot*

Ahora debería en el Vcenter aparecer la IP y el DNS.

Debemos cambiar en .ssh de windows en known\_hosts para la IP de la máquina y hacemos ping y cambiamos la pass de root con passwd.

mkdir -p /usr/local/pr/cassandra/

groupadd cassandra

useradd cassandra -g cassandra -d /usr/local/pr/cassandra/

chown cassandra:cassandra -R /usr/local/pr/cassandra/

ls -la /usr/local/pr/cassandra/

passwd cassandra

source .bashrc

* 1. **Instalación y configuración de Ansible**
     1. **Creación de playbooks de Ansible**

Para la instalación de los diferentes aplicativos de la transformación se han creado los siguientes roles de Ansible ( dentro de scripts/ansible-playbooks ) que instalan diferentes paquetes para cada tipo de instancia diferente:

- common ( Role que instala las dependencias comunes a todas las máquinas y que se debe ejecutar inicialmente en todas )

- registry ( Role que instala las dependencias y aplicaciones para el Registry de Docker )

- k8s ( Role que instala las dependencias y aplicaciones para los nodos workers y masters de Kubernetes )

- etcd ( Role que instala las dependencias y aplicaciones para los nodos etcd )

- bastión ( Role que instala las dependencias y aplicaciones para los nodos bastión de Kubernetes )

- balanceador ( Role que instala las dependencias y aplicaciones para los balanceadores de Kubernetes )

Para ejecutar estos roles sobre una máquina con Ansible instalado debemos tener la VPN de CIN activa y ejecutar el siguiente comando para ejecutar un determinado role:

`ansible-playbook -i inventario --limit <máquina> sites.yaml`

Adicionalmente, se pensó en descargar los paquetes necesarios para cada role desde el Drive y meterlos en el directorio local donde se ejecutaría Ansible ( dentro del role en el directorio files).

Pero finalmente podremos apuntar al repo oficial de RHEL 8.6, por tanto, se descargarán los paquetes a instalar (latest) de dicho repo para su posterior instalación.

Esto ejecutará el role de common en la máquina especificada en el parámetro limit y después el role determinada por el inventario de Ansible ( \*\*scripts/ansible-playbooks/inventario\*\* ) . Adicionalmente se puede ejecutar una determinada tarea especificada según la tag que se defina en las tasks de ese role:

`ansible-playbook -i inventario --limit <máquina> --tags <tag> sites.yaml`

\*\*Configurar Clientes\*\*

Para conectarse a este Registry es necesario importar el certificado \*\*loadbalancer.cin.cert\*\* del Registry ( [loadbalancer.cin.cert](https://drive.google.com/file/d/1qlqy7vXCh2HvbGDUhyHSQRyZeF7JKvMe/view?usp=sharing) ) a la ruta /\*\*etc/docker/certs.d/loadbalancer.cin\*\* ( es necesario crearla ) y después reiniciar el servicio de Docker.

Añadimos de forma temporal el host al fichero de hosts:

\*\*echo "192.168.112.139 loadbalancer loadbalancer.cin" >> /etc/hosts\*\*

Después nos logueamos contra el Registry ( usuario admin) :

\*\*docker login loadbalancer.cin:443\*\*

Para bajarnos imágenes de Internet mediante el proxy:

\*\*docker pull loadbalancer.cin:443/proxy\_docker\_hub/bitnami/openldap:2.5.11\*\*

O bien de Kubernetes (repositorio k8s.gcr.io) :

\*\*docker pull loadbalancer.cin:443/k8s.gcr.io/pause:3.5\*\*

\*\*docker pull loadbalancer.cin:443/k8s.gcr.io/coredns/coredns:v1.8.4\*\*

Se puede observar que el Registry ha accedido mediante el proxy al repositorio de Internet k8s.gcr.io y ha bajado las imágenes requeridas.

* + 1. **Definición de las tareas y roles necesarios**
    2. **Configuración de los hosts y los grupos**
    3. **Creación de variables y templates**
    4. **Instalación de common necesarios**
    5. **Despliegue del clúster de Kubernetes**

Instalación de containerd y nerdctl\*\*

\*\*Descripción:\*\* se instala containerd y nerdctl con todo lo que necesitan, y se realiza una prueba de levantar contenedor de nginx.

1 - Extraer contenido del .tar.gz de nerdct-fulll (binarios, librerías, etc) en /usr/local/pr/kamino.

tar Cxzvvf /usr/local/pr/kamino nerdctl-full-0.18.0-linux-amd64.tar.gz

2 - Generamos el config.toml con los siguientes comandos:

sudo mkdir -p /etc/containerd/config.toml

containerd config default > /etc/containerd/config.toml

3 - Arrancamos el servicio de containerd :

sudo systemctl enable --now containerd

Esto genera el fichero /usr/usr/local/lib/systemd/system/containerd.service

4 - Seguidamente especificamos este archivo de configuración de containerd en el daemon de containerd:

sudo vi /usr/local/lib/systemd/system/containerd.service

ExecStart=/usr/local/bin/containerd -c /etc/containerd/config.toml

5 - Configurar servicio containerd para que el directorio de almacenamiento persistente (pods, addons y plugins, etc). Se modifica con el parámetro root y state al iniciar el servicio con systemd (alternativamente se puede crear un archivo de configuración, ver https://github.com/containerd/containerd/blob/main/docs/ops.md#:~:text=%2D%2Droot%20 value%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20containerd %20root%20directory , archivo /etc/containerd/config.toml https://containerd.io/docs/getting-started/ ).

root = "/usr/local/pr/kamino/var/lib/containerd" state = "/usr/local/pr/kamino/run/containerd"

6 - Recargamos systemd:

systemctl daemon-reload

7 - Reiniciamos el servicio de containerd:

systemctl restart containerd

8- Test: levantar contenedor de alpine

nerdctl run --rm -it alpine:latest

\*\*Paquetes:\*\* nerdctl-full-0.14.0-linux-amd64.tar.gz , nginx-alpine.tar

\*\*Scripts:\*\* install-containerd-nerdctl.sh

* + 1. **Instalación de los componentes del clúster (master y etcd, workers, bastión, registry, loadbalancer, etcd)**

Instalación y configuración de Docker y Harbor, Creación y configuración del Registry\*\*

Se ha creado un role de Ansible que instala Docker y sus dependencias , seguidamente lo configura para que la ruta donde crea los contenedores sea /usr/local/pr/kamino/var/lib/docker/overlay2. Para las pruebas se ha utilizado la máquina del loadbalancer ( 192.168.112.139 ) aunque finalmente se llamará registry.cin o similar.

Seguidamente, se instala Harbor cargando las imágenes de Docker forma offline y se configura mediante la creación de una CA custom la cual firma la clave privada y certificado y asigna al Nginx que levanta harbor. Se ha preconfigurado el instalador para que instale Trivy , chartmuseum y Harbor de forma offline sin que accedan a Internet salvo para descargar nuevas imágenes no presentes ( install.sh --with-trivy --with-chartmuseum ).

Se pueden ver los ficheros de instalación en el repositorio siguiente de Gitlab => /transformacion-kamino/instalacion-k8s/-/tree/master/scripts/ansible-playbooks/kubernetes-installation/roles/registry]

Si se quiere ejecutar el playbook se deben bajar los ficheros siguientes a la ruta roles/registry/files para que Ansible pueda realizar la instalación de este role correctamente.

Al menos son necesarios los siguientes ficheros en la ruta roles/registry/files:

- \*\*daemon.json\*\* ( Fichero de configuración de Docker para confiar en el Registry )

- \*\*docker-ce-offline.tar.gz\*\* ( Fichero comprimido con las dependencias de Docker)

- \*\*docker-compose.yml\*\* ( compose con el stack de Harbor )

- \*\*harbor-offline-installer-v2.4.2.tgz\*\* ( Fichero comprimido con todas las imágenes de Harbor )

- \*\*harbor.yml\*\* ( Fichero que lee el instalador de harbor ( install.sh ) para arrancar el entorno por primera vez )

- \*\*install.sh\*\* ( Script que lee el fichero harbor.yml y prepara el entorno de Harbor , y comprueba que Docker y docker-compose estén previamente instalados )

- \*\*metadata.json\*\* ( Fichero de configuración para trivy para que no se actualice )

- \*\*trivy.db\*\* ( Base de datos de CVEs de las imágenes de Docker )

- \*\*v3.ext\*\* ( x509 v3 extension file para la creación del certificado del Registry )

Los ficheros se pueden encontrar en la siguiente carpeta del Drive.

Para ejecutar este playbook bajamos el repositorio de Gitlab mencionado anteriormente en una máquina con Ansible instalado ( con Python 3.5 o superior ) , después bajamos los ficheros mencionados anteriormente en la ruta roles/registry/files y lo ejecutamos mediante los siguientes comandos:

Si le ponemos las tags docker-install y harbor-install solo realizará la instalación de Docker y Harbor :

\*\*ansible-playbook -i inventario --limit harbor --ask-vault-pass --tags=”docker-install” sites.yaml\*\*

\*\*ansible-playbook -i inventario --limit harbor --ask-vault-pass --tags=”harbor-install” sites.yaml\*\*

Si se desea realizar una instalación completa, primero de los paquetes comunes y después de Docker y Harbor se debe ejecutar de la siguiente manera:

\*\*ansible-playbook -i inventario --limit harbor --ask-vault-pass sites.yaml\*\*

De esta manera también es necesario bajar los ficheros comunes ( roles/common/files) del Drive al directorio local de nuestro PC con Ansible.

URL Registry Actual Funcionando =>

[https://192.168.112.139/account/sign-in?redirect\_url=%2Fharbor%2Fprojects](https://192.168.112.139/account/sign-in?redirect\_url=%2Fharbor%2Fprojects)

(Credenciales las mismas que el Registry actual de Producción )

* + 1. **Configuración de los nodos y los componentes**

1. **Resultados**
   1. **Verificación del funcionamiento del clúster (gestionado con Rancher Desktop)**
   2. **Demostración**

Como resultado de este proyecto tendremos:

* Repositorio en GitHub con el código fuente
* Plantilla de Cookiecutter para la personalización del despliegue
* Repositorio con las dependencias necesarias para operar offline
* Demostración del clúster, recursos y funcionalidades
* Documento con el análisis y diseño propuestos, así como con los pasos seguidos para la elaboración

Por ejemplo, un vídeo demostrando métricas y gráficas, carga de CPU y recursos (Prometheus + grafana + kibana).

Creo que lo más llamativo de la idea es el tema de la automatización de la instalación del clúster con Ansible, ya que facilita mucho las cosas y puedes replicar en distintas máquinas que tengas en el inventario. Por tanto, mostrar con AWX (interfaz de Ansible<https://github.com/ansible/awx>) la instalación del clúster.

Gestión del clúster con K3S y Rancher.

Quizá también mostrando un frontal de algún aplicativo montado o una demo como la de<https://microservices-demo.github.io/>

1. **Conclusiones**
   1. **Ventajas de utilizar Ansible para montar un clúster de Kubernetes**
2. **Futuras tendencias y vista a futuro**
   1. **Otras formas de instalar en k8s con Helm**
3. **Bibliografía**
4. Valladares, C. S. T., & Sacoto, A. S. Q. (2022). Procesos de protección en entornos de ejecución de contenedores Kubernetes para una entidad financiera: una revisión sistemática. Dominio de las Ciencias, 8(4), 619-644.
5. Jens Erat, Peter Mueller, Sabine Wolz. [CNCF [Cloud Native Computing Foundation]] (2 jun 2022). *7 Years of Running Kubernetes for Mercedes-Benz* [Video]. Youtube. <https://www.youtube.com/watch?v=UmbjwSK9b3I&list=PLj6h78yzYM2PfD9vkHopnzNNIVicOFtih&index=18>
6. Kubernetes,<https://kubernetes.io/>

**DESCRIPCIÓN DE LAS TAREAS**

Con el fin de llegar al objetivo final de este proyecto, la instalación automatizada del clúster, se llevarán a cabo diversas tareas.

En primer lugar, debemos familiarizarnos con estas nuevas tecnologías. Para ello habrá que seguir los siguientes pasos:

* Desplegar un entorno de orquestación de contenedores llevando a cabo la configuración necesaria.
* Al mismo tiempo que se elaboran estas tareas, se efectuará la tarea de las pruebas, muy importante en el ciclo de desarrollo de software.

Con la tecnología entendida e instalada, es decir, ya con la infraestructura en funcionamiento, se comprobará si el funcionamiento es el esperado. Por tanto:

* Estudiar que el funcionamiento es el correcto tras todo el despliegue.
* Tras el análisis se llevará a cabo una conclusión de los resultados y de lo aprendido en el desarrollo del proyecto.
* Finalmente, se desarrollará un video en forma de demo el cual servirá para demostrar las capacidades de todo lo desarrollado.

Esquema de bloques:

Diagrama

Descripción generada automáticamente

**METODOLOGÍA Y PLAN DE TRABAJO**

Para proceder a la realización del trabajo de fin de grado se ha optado por dividirlo en diferentes fases para poder llevar un control y seguimiento de las tareas de manera específica. El total de horas de trabajo será de 350 aproximadamente.

El trabajo se ha dividido en semanas y no en días para una mayor libertad en la consecución de los objetivos. Cada semana comprenderá 17.5 horas de trabajo.

De cara a la consecución de los objetivos del proyecto que se han descrito anteriormente, estas serán las fases a seguir:

**1. Formación inicial (5 semanas):**

* Formación en el uso de Ansible, Docker y Kubernetes (3 semanas): Durante esta fase, se obtendrán los conocimientos que permitan un manejo óptimo de la herramienta. Para ello se tomarán como ayuda la documentación proporcionada por la propia herramienta y las guías de inicio.
* Consulta bibliográfica y de la API (2 semanas): Recopilación de la información necesaria tanto de la comunidad como de la API.

**2. Análisis y diseño del entorno (4 semanas):**

* Análisis de los requisitos y arquitecturas(4 semanas): Se llevará a cabo un análisis de los requerimientos de cada una de las máquinas que formarán el clúster con planteamiento de las arquitecturas que sean necesarias para la posterior implementación de estas.

**3. Implementación y desarrollo de las configuraciones (10 semanas):**

* Desarrollo del código fuente (scripts y playbooks) (6 semanas): Durante esta etapa, se identificarán las dependencias necesarias, se elegirán las versiones adecuadas y se escogerán los roles necesarios. Tras esto, se procederá a materializar todo el conocimiento en el código.
* Pruebas (3 semanas): En esta fase se busca detectar los fallos que se han podido cometer en etapas anteriores y corregirlos.
* Correcciones estéticas (0,5 semanas): Retoques y modificaciones a nivel decorativo y artístico.
* Documentación del código (0,5 semanas): Comentar y documentar de forma adecuada el código de los diferentes programas.

**4. Documentación y finalización del proyecto (1 semanas):** La documentación del proyecto se irá realizando a lo largo de toda la planificación y en la última semana se pondrá especial atención en los últimos detalles.

