

**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

Resumen

El cloud computing, es un modelo tecnológico que se está implementando en la mayoría de empresas en los últimos años. Sus ventajas son obvias, ya sea porque se usa Software como servicio (SaaS), Plataforma como servicio (PaaS) o Infraestructura como servicio (IaaS).

La computación en la nube se puede categorizar según los métodos de entrega, que depende de la aplicación informática, la plataforma y el hardware que se ofrezcan. Estos modelos son:   
- Software como servicio (SaaS): el proveedor de servicios en la nube aloja el software demandado en sus servidores, haciendo que los usuarios solo tengan que preocuparse por cómo usar ese software, olvidándose de las capas inferiores.

- Plataforma como servicio (PaaS): el proveedor de servicios en la nube ofrece una plataforma sobre la que desplegar aplicaciones. En estas plataformas se pueden lanzar desde herramientas de desarrollo hasta middleware o bases de datos. El cliente solo se hará cargo del desarrollo y despliegue de aplicaciones sin preocuparse de la infraestructura inferior.

- Infraestructura como servicio (IaaS): el cliente contrata una infraestructura de hardware al proveedor de servicios de computación en la nube. Esto permite elegir detalladamente los requisitos de capacidad de proceso, memoria a utilizar o espacio de almacenamiento.

Ello supone para las empresas un ahorro de costes, espacios seguros, entornos accesibles y una personalización del servicio según los requisitos del cliente.

En este trabajo se va a llevar a cabo la instalación y configuración de la plataforma de orquestación de contenedores Kubernetes a través de un clúster, en este caso, sobre un conjunto de máquinas virtuales. El objetivo es conseguir una plataforma lo más semejante posible a aquellas utilizadas en entornos corporativos. Se pretende aprender sobre el uso de esta tecnología y sobre su implementación en un entorno real, así como conceptos tales como la computación en la nube, contenedores o virtualización.  
   
La realización del proyecto contempla también automatizar todo este proceso de tal manera que se supriman la mayor cantidad de pasos manuales posibles y se optimicen así los tiempos de los procesos gracias a herramientas de infraestructura como código.

Otra ventaja de la automatización es el hecho de que, al eliminar pasos en los que el usuario ha de realizar operaciones manualmente, es menos susceptible a fallos que una instalación completamente realizada a mano.

El objeto de este trabajo es estudiar qué facilidades, herramientas y comodidades se ofrecen a la hora de desplegar un clúster de Kubernetes altamente disponible de forma automatizada.

Abstract

Cloud computing is a technological model that is being implemented in most companies in recent years. Its advantages are obvious, whether it is Software as a Service (SaaS), Platform as a Service (PaaS) or Infrastructure as a Service (IaaS).

Cloud computing can be categorised according to the delivery methods, which depends on the software application, platform and hardware offered. These models are:

- Software as a Service (SaaS): the cloud service provider hosts the demanded software on its servers, making users only have to worry about how to use that software, forgetting about the lower layers.

- Platform as a Service (PaaS): the cloud service provider offers a platform on which to deploy applications. These platforms can be used to launch everything from development tools to middleware or databases. The customer is only responsible for the development and deployment of applications without worrying about the underlying infrastructure.

- Infrastructure as a Service (IaaS): the customer contracts a hardware infrastructure from the cloud computing service provider. This allows a detailed choice of processing capacity, memory or storage space requirements.

For companies, this means cost savings, secure spaces, accessible environments and customisation of the service according to the customer's requirements.

In this work we are going to carry out the installation and configuration of the Kubernetes container orchestration platform through a cluster, in this case, on a set of virtual machines. The aim is to achieve a platform as similar as possible to those used in corporate environments. The aim is to learn about the use of this technology and its implementation in a real environment, as well as concepts such as cloud computing, containers or virtualisation.

The project also aims to automate this entire process in such a way that as many manual steps as possible are eliminated and process times are optimised thanks to infrastructure tools such as code.

Another advantage of automation is the fact that, by eliminating steps where the user has to perform operations manually, it is less susceptible to failures than a completely manual installation.

The purpose of this paper is to study what facilities, tools and conveniences are offered when deploying a highly available Kubernetes cluster in an automated way.

Tabla de contenidos

[**1. Introducción y objetivos 4**](#_heading=h.9w22495yx7dc)

[1.1. Introducción 4](#_heading=h.h7kg00706r86)

[1.2. Motivación 5](#_heading=h.tjizwdsqt3ok)

[1.3. Objetivo 6](#_heading=h.ch6bpjgq59ta)

[1.4. Estructura del documento 7](#_heading=h.dfu0mmdgtb69)

[**2. Antecedentes 8**](#_heading=h.17y9zgmbws9e)

[2.1. Contenedores 8](#_heading=h.eh6w2v2lk06j)

[2.2. Kubernetes 8](#_heading=h.908drzve7kr)

[2.3. Ansible 8](#_heading=h.6buwha6sr9nc)

[**3. Desarrollo 8**](#_heading=h.yz2onibu7zpu)

[3.1. Prerrequisitos y preparación del entorno 8](#_heading=h.d0sca4c7u4q)

[3.2. Planificación de uso de recursos (Requisitos de hardware y software) 9](#_heading=h.dr2a4xichbx7)

[3.2.1. Creación de máquinas independientes (setup-host.sh) 9](#_heading=h.2uvas3emjsmh)

[3.3. Instalación y configuración de Ansible 15](#_heading=h.9se4wag6wcus)

[3.3.1. Creación de playbooks de Ansible 15](#_heading=h.bmkfz8ee42mr)

[3.3.2. Definición de las tareas y roles necesarios 18](#_heading=h.9qighj75ng5p)

[3.3.3. Configuración de los hosts y los grupos 18](#_heading=h.k5xt8k8uyq31)

[3.3.4. Creación de variables y templates 18](#_heading=h.wp8jpcmo2smf)

[3.3.5. Instalación de common necesarios 18](#_heading=h.4l7rhoh4w7hm)

[3.3.6. Despliegue del clúster de Kubernetes 18](#_heading=h.bjfouki1mw1n)

[3.3.7. Instalación de los componentes del clúster (master/etcd, workers, bastión, registry, loadbalancer) 19](#_heading=h.p3svz46phbey)

[3.3.8. Configuración de los nodos y los componentes 21](#_heading=h.dvq351cd4743)

[**4. Resultados 21**](#_heading=h.q4qyvqc928qm)

[4.1. Verificación del funcionamiento del clúster (gestionado con Rancher Desktop) 22](#_heading=h.drqe50k6nux7)

[4.2. Demostración 22](#_heading=h.4m5hz3h8z5di)

[**5. Conclusiones 22**](#_heading=h.r68nxex76oxo)

[5.1. Ventajas de utilizar Ansible para montar un clúster de Kubernetes 22](#_heading=h.guenv03ft7g7)

[**6. Futuras tendencias y vista a futuro 22**](#_heading=h.2plroee7sbnw)

[6.1. Otras formas de instalar en k8s con Helm 22](#_heading=h.qgx9m0qrdo48)

[**7. Bibliografía 22**](#_heading=h.xubeq6zbztv2)

# Introducción y objetivos

### Introducción

En julio de 2022 comencé en una empresa dedicada principalmente al sector aeroespacial y consultoría, más concretamente en la sección de SecDevOps and Security Monitoring. Durante el periodo de prácticas me dediqué a formarme en tecnologías Cloud Native y relacionadas con esta. Tras unos meses aprendiendo y trabajando en un proyecto con cliente de sector bancario surgió la idea de crear un pequeño entorno de laboratorio en el que se pudieran hacer todo tipo de pruebas sin compromiso y que más tarde sirviese también como entorno de aprendizaje para las personas que fuesen llegando y formándose en Ansible y Kubernetes. Idea motivada también por un proceso de transformación de infraestructura en dicha entidad, el cual, estamos llevando a cabo en el proyecto.

Tras la propuesta de mi tutor de empresa sobre la idea de crear un laboratorio desde mi proyecto de fin de grado comencé a investigar y reunir los requisitos necesarios para llevarlo a cabo.

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.

### Motivación

La virtualización de sistemas e infraestructuras en empresas es una práctica cada vez más común e interesante, por tanto, decidí que esta temática era muy interesante y que apenas se había profundizado durante el Grado.

Aprovechando al máximo unos recursos limitados, se plantea desplegar un entorno con un mantenimiento y gestión de aplicaciones pensado de la forma más sencilla posible.

Antiguamente la solución más utilizada en torno a la infraestructura era alojar tus servicios y aplicaciones en un solo ordenador físico. Es por esto que, podían existir problemas relacionados con las versiones de las herramientas que utilizaba cada programa y más lentitud a la hora de desplegar nuevas soluciones y código de los desarrolladores.

Posteriormente, la tecnología evolucionó y dio paso a las máquinas virtuales. Esto solucionaba las incompatibilidades antes mencionadas, ya que las máquinas virtuales pueden crear entornos aislados para albergar diferentes servicios. Sin embargo, las máquinas virtuales traen otros problemas, como el uso excesivo de recursos.

Los contenedores aparecieron para resolver estos problemas y proporcionar una virtualización ligera. Nos aportan el entorno mínimo necesario para la aplicación encargada de prestar el servicio y ahorra en gran medida los recursos de las máquinas físicas, ya que los servidores continúan ejecutándose en entornos separados, pero estos están específicamente optimizados.

Por tanto, la motivación de este proyecto viene dada por esta serie de factores que llevan a la búsqueda de una solución capaz de coordinar múltiples máquinas prestando un servicio con una de las herramientas que cada vez más empresas están adoptando en sus entornos productivos como es Kubernetes. Además, se mostrarán sus ventajas e inconvenientes así como los mejores escenarios para utilizarla.

### 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.

Se pretende que los escenarios finales sean iguales o muy similares a los utilizados por las empresas que usan tales soluciones, y comprenda cómo estos entornos difieren de otros entornos más pequeños, como el entorno mínimo recomendado por Red Hat.

Por todo ello, el objetivo final es poder proporcionar un entorno completamente funcional en el que se puedan realizar pruebas con diversos riesgos. Dada la posibilidad de que un clúster se vuelva completamente inutilizable, aquí es donde entra en juego la automatización. Se busca un entorno que se pueda configurar en el menor tiempo posible con la menor cantidad de trabajo.

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

Para lograr todo esto debemos familiarizarnos con estas nuevas tecnologías y habrá que seguir los siguientes objetivos paso a paso:   
- Aprender sobre las tecnologías de contenedores.   
- Aprender sistemas de uso de contenedores como Docker.   
- Aprender sobre el uso de herramientas de virtualización.  
- Aprender sobre el uso de herramientas de automatización e infraestructura como código.  
- Conocer las ventajas de los contenedores sobre otras tecnologías.  
- Aprender sobre sistemas de orquestación de contenedores, en este caso Kubernetes. - Identificar las ventajas del uso de estas plataformas orquestadoras.

Tras haber logrado entender el funcionamiento de estas tecnologías habrá que descubrir las posibles formas de instalar estas plataformas. Por ello habrá que:  
- Aprender a desplegar un entorno de orquestación de contenedores.  
- Aprender a usar la terminal o el entorno gráfico para llevar a cabo la configuración necesaria.

Con la tecnología entendida e instalada, es decir, ya con la infraestructura en funcionamiento, se comprobará si el funcionamiento es el esperado. Así hay que:  
- Estudiar que el funcionamiento es el correcto tras todo el despliegue.

- Desarrollo de un video en forma de demo para demostrar las capacidades de todo lo desarrollado.

- Realizar las conclusiones de todo lo analizado.

### Estructura del documento

La organización del documento se basa en la creación de diferentes secciones con información diversa para cada uno de ellos. Estas secciones a su vez estarán compuestas de subapartados para dividir la información de manera que el documento sea más manejable y proporcione una mayor legibilidad.  
   
En cuanto al contenido, a esta página le seguirá el apartado “Antecedentes” en el que se busca explicar todo lo que ha sido importante para la realización del trabajo y las tecnologías que ya existían cuando comenzó el desarrollo. El objetivo es presentar al lector algunos conceptos que son esenciales e importantes para comprender el desarrollo del proyecto.   
Le sigue el apartado "Desarrollo", que proporciona una descripción informatizada del proyecto, explicando el trabajo realizado y los diferentes puntos para separar la información mostrada.  
   
Hacia el final del documento se encontrará el apartado “Resultados”, en el que se harán una serie de pruebas sobre el proyecto y se mostrarán los resultados obtenidos de la realización de este. Seguidamente se planteará el apartado de “Conclusiones”, en el que se realizará una valoración personal del trabajo hecho y se justificará si se han conseguido los objetivos planteados. Se analizarán también las limitaciones del trabajo y se propondrán posibles formas de continuarlo a futuro.

Para finalizar contaremos con el apartado “Bibliografía”, en el que se recogerán en formato APA los recursos utilizados como referencia a lo largo del desarrollo del proyecto.

# Antecedentes

### Contenedores

Históricamente, a medida que aumentaba la potencia y capacidad de procesamiento de los servidores, las aplicaciones bare metal no eran capaces de explotar la nueva abundancia de recursos. Así nacieron las máquinas virtuales, diseñadas mediante la ejecución de software sobre servidores físicos para emular un sistema de hardware concreto. Un hipervisor, o monitor de máquina virtual, es el software, firmware o hardware que crea y ejecuta las máquinas virtuales. Es lo que se sitúa entre el hardware y la máquina virtual y es necesario para virtualizar el servidor.

Dentro de cada máquina virtual se ejecuta un único sistema operativo invitado. En el mismo servidor físico pueden ejecutarse máquinas virtuales con sistemas operativos diferentes: una máquina virtual UNIX puede estar junto a una máquina virtual Linux, y así sucesivamente. Cada máquina virtual tiene sus propios binarios, bibliotecas y aplicaciones a las que da servicio, y la máquina virtual puede tener un tamaño de muchos gigabytes.

La virtualización de servidores ofrece una serie de ventajas, una de las más importantes es la posibilidad de consolidar las aplicaciones en un único sistema. Atrás quedaron los días en que una sola aplicación se ejecutaba en un único servidor. La virtualización supuso un ahorro de costes gracias a la reducción de la huella, el aprovisionamiento más rápido de servidores y la mejora de la recuperación ante desastres (DR), porque el hardware del sitio de DR ya no tenía que reflejar el centro de datos primario.

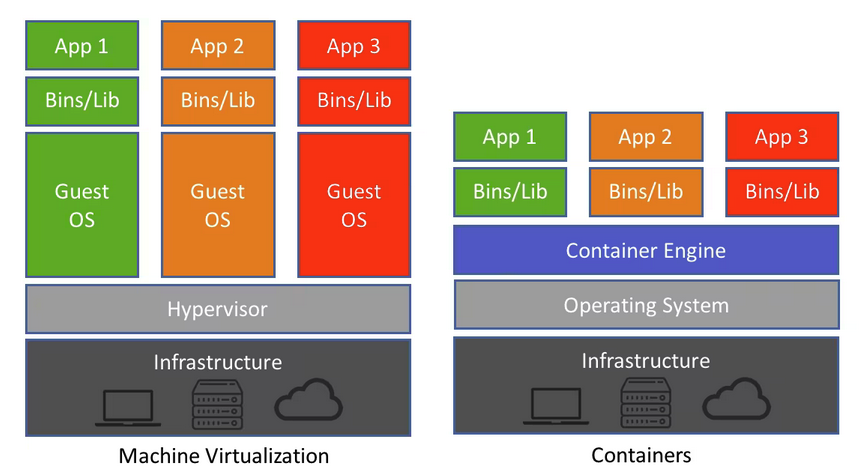
El desarrollo también se benefició de esta consolidación física, ya que la mayor utilización de servidores más grandes y rápidos liberó servidores que no se utilizaban posteriormente para reutilizarlos como equipos de control de calidad, desarrollo o laboratorio. Pero este enfoque ha tenido sus inconvenientes. Cada máquina virtual incluye una imagen independiente del sistema operativo, lo que añade sobrecarga de memoria y almacenamiento. Resulta que este problema añade complejidad a todas las fases del ciclo de vida del desarrollo de software, desde el desarrollo y las pruebas hasta la producción y la recuperación ante desastres. Este enfoque también limita en gran medida la portabilidad de las aplicaciones entre nubes públicas, nubes privadas y centros de datos tradicionales. ¿Qué son los contenedores? La virtualización del sistema operativo (SO) ha ganado popularidad en la última década para permitir que el software se ejecute de forma predecible y correcta cuando se traslada de un entorno de servidor a otro. Pero los contenedores ofrecen una forma de ejecutar estos sistemas aislados en un único servidor o sistema operativo anfitrión.

Los contenedores se sitúan sobre un servidor físico y su sistema operativo anfitrión, por ejemplo, Linux o Windows. Cada contenedor comparte el núcleo del sistema operativo anfitrión y, normalmente, también los binarios y las bibliotecas. Los componentes compartidos son de sólo lectura. Los contenedores son, por tanto, excepcionalmente "ligeros": sólo ocupan megabytes y tardan segundos en iniciarse, frente a los gigabytes y minutos de una máquina virtual.

Los contenedores también reducen la sobrecarga de gestión. Dado que comparten un sistema operativo común, sólo un sistema operativo necesita cuidados y alimentación para correcciones de errores, parches, etcétera. Este concepto es similar al que experimentamos con los hosts de hipervisor: menos puntos de gestión pero un dominio de fallos ligeramente superior. En resumen, los contenedores son más ligeros y portátiles que las máquinas virtuales. Conclusión Las máquinas virtuales y los contenedores difieren en varios aspectos, pero la principal diferencia es que los contenedores proporcionan una forma de virtualizar un sistema operativo para que puedan ejecutarse varias cargas de trabajo en una única instancia del sistema operativo. Con las máquinas virtuales, el hardware se virtualiza para ejecutar varias instancias del sistema operativo. La velocidad, agilidad y portabilidad de los contenedores los convierten en una herramienta más para agilizar el desarrollo de software.

En resumen, los beneficios de usar contenedores incluyen:

* **Ágil creación y despliegue de aplicaciones**: Mayor facilidad y eficiencia al crear imágenes de contenedor en vez de máquinas virtuales
* **Desarrollo, integración y despliegue continuo**: Permite que la imagen de contenedor se construya y despliegue de forma frecuente y confiable, facilitando los rollbacks pues la imagen es inmutable
* **Separación de tareas entre Dev y Ops**: Puedes crear imágenes de contenedor al momento de compilar y no al desplegar, desacoplando la aplicación de la infraestructura
* **Observabilidad** No solamente se presenta la información y métricas del sistema operativo, sino la salud de la aplicación y otras señales
* **Consistencia entre los entornos de desarrollo, pruebas y producción**: La aplicación funciona igual en un laptop y en la nube
* **Portabilidad entre nubes y distribuciones**: Funciona en Ubuntu, RHEL, CoreOS, tu datacenter físico, Google Kubernetes Engine y todo lo demás
* **Administración centrada en la aplicación**: Eleva el nivel de abstracción del sistema operativo y el hardware virtualizado a la aplicación que funciona en un sistema con recursos lógicos
* **Microservicios** distribuidos, elásticos, liberados y débilmente acoplados: Las aplicaciones se separan en piezas pequeñas e independientes que pueden ser desplegadas y administradas de forma dinámica, y no como una aplicación monolítica que opera en una sola máquina de gran capacidad
* **Aislamiento de recursos**: Hace el rendimiento de la aplicación más predecible
* **Utilización de recursos**: Permite mayor eficiencia y densidad



*Fig. 1. Máquinas virtuales contra contenedores.*

Como podemos observar en la figura, las VM virtualizan el hardware y cada una lleva su propio sistema operativo por encima del hipervisor y el sistema operativo anfitrión. En cambio, los contenedores virtualizan el sistema operativo, cada uno tiene su CPU, memoria y almacenamiento, pero comparten el mismo kernel con los demás contenedores en el anfitrión.

### Kubernetes

Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios facilitando la automatización y la configuración declarativa. Tiene un ecosistema grande y en rápido crecimiento.

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

Se puede pensar en Kubernetes como:

* una plataforma de contenedores
* una plataforma de microservicios
* una plataforma portable de nube

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

A pesar de que Kubernetes ya ofrece muchas funcionalidades, siempre hay nuevos escenarios que se benefician de nuevas características. Los flujos de trabajo de las aplicaciones pueden optimizarse para acelerar el tiempo de desarrollo. Una solución de orquestación propia puede ser suficiente al principio, pero suele requerir una automatización robusta cuando necesita escalar. Es por ello que Kubernetes fue diseñada como una plataforma para poder construir un ecosistema de componentes y herramientas que hacen más fácil el desplegar, escalar y administrar aplicaciones.

Los objetos son entidades duraderas dentro de Kubernetes, cuando creas un objeto en Kubernetes, debes especificar la “spec” del objeto, que describe su estado deseado, así como información básica del mismo. Cuando usas la API de Kubernetes para crear el objeto, dicha petición a la API debe incluir toda la información en formato JSON. Esta información es proporcionada a través de manifiestos, archivos de tipo .yaml, kubectl convierte esa información a JSON cuando realiza la llamada a la API.

Un clúster de Kubernetes se divide en nodos, máquinas con el software necesario ya instalado para ejecutar las operaciones requeridas. Estás máquinas están controladas por una máquina llamada “master” o maestra. En ella es dónde se origina la asignación de tareas.

Los contenedores son agrupados en pods, la unidad operativa mínima y más pequeña de Kubernetes. Los pods añaden una capa de abstracción a estos grupos de contenedores. Todos los contenedores de un pod comparten la dirección IP, el nombre del host y otros recursos. Los pods abstraen la red y el almacenamiento del contenedor subyacente. Esto permite mover los contenedores por el clúster con mayor facilidad.

Los pods cuentan por defecto con almacenamiento efímero, de modo que el pod contará con un almacenamiento que se eliminará cuando el pod sea eliminado o muera. Además, el almacenamiento efímero no puede ser compartido por pods, de manera que dos pods que alojan a una misma aplicación necesitarán cambiar el almacenamiento por defecto, efímero, a un almacenamiento persistente del que varios clientes puedan leer y escribir a la vez, por ejemplo, aquel ofrecido por un servidor NFS.

Para esto existen dos tipos de recursos de Kubernetes llamados “Persistent Volume” y “Persistent Volume Claim”, referidos habitualmente como “PV” y “PVC” respectivamente. Un PV es una pieza de almacenamiento en el clúster que ha sido aprovisionada por un administrador o dinámicamente. Es un recurso en el cluster al igual que un nodo es un recurso del clúster. Los PVs tienen un ciclo de vida independiente de cualquiera que los utilice. Este objeto API captura los detalles de la implementación del almacenamiento, ya sea NFS, iSCSI, o un sistema de almacenamiento específico del proveedor de la nube, en este caso VMWare. Un PVC es una solicitud de almacenamiento por parte de un usuario en la que se pueden especificar parámetros como el tamaño de almacenamiento o el modo de acceso a este.

Existen también los llamados servicios, objetos que sirven para exponer una aplicación que se ejecuta en un conjunto de pods como un servicio de red. Ofrecen una dirección virtual y un nombre que identifica al conjunto de pods que representa, al cual nos podemos conectar. La conexión al servicio se puede realizar desde otros pods o desde el exterior.

También cabe destacar los controladores de replicación, uno de los objetos que le brinda a Kubernetes una de sus funcionalidades más potentes. Como su propio nombre indica, controla la cantidad de copias que tiene que haber en el clúster de un pod. Esto es utilizado para la autorregeneración de aplicaciones con ubicación, reinicio, replicación y escalamiento automáticos.

### Ansible

Ansible es una herramienta de software que proporciona una automatización sencilla pero potente para el soporte informático multiplataforma. Está destinada principalmente a profesionales de TI, que la utilizan para el despliegue de aplicaciones, actualizaciones en estaciones de trabajo y servidores, aprovisionamiento en la nube, gestión de la configuración, orquestación intraservicio y casi cualquier cosa que un administrador de sistemas haga semanal o diariamente. Ansible no depende de software de agente y no tiene infraestructura de seguridad adicional, por lo que es fácil de desplegar.

Dado que Ansible se basa en la automatización, requiere instrucciones para realizar cada tarea. Con todo escrito en forma de script simple, resulta sencillo aplicar un control de versiones. El resultado práctico de esto es una importante contribución al movimiento "infraestructura como código" en TI: la idea de que el mantenimiento de la infraestructura de servidores y clientes puede y debe tratarse igual que el desarrollo de software, con repositorios de soluciones autodocumentadas, probadas y ejecutables capaces de hacer funcionar una organización independientemente de los cambios de personal.

Aunque Ansible puede estar a la vanguardia de la automatización, la administración de sistemas y DevOps, también es útil para los usuarios cotidianos. Ansible permite configurar no sólo un ordenador, sino potencialmente toda una red de ordenadores a la vez, y su uso no requiere conocimientos de programación. Las instrucciones escritas para Ansible son legibles. Tanto si eres un novato en informática como un experto, los archivos de Ansible son fáciles de entender.

En Ansible, hay dos categorías de ordenadores: el nodo de control y los nodos gestionados. El nodo de control es un ordenador que ejecuta Ansible. Debe haber al menos un nodo de control, aunque también puede existir un nodo de control de respaldo. Un nodo gestionado es cualquier dispositivo gestionado por el nodo de control.

Ansible funciona conectándose a nodos (clientes, servidores, o lo que sea que estés configurando) en una red, y luego enviando un pequeño programa llamado módulo Ansible a ese nodo. Ansible ejecuta estos módulos a través de SSH y los elimina cuando termina. El único requisito para esta interacción es que el nodo de control de Ansible tenga acceso a los nodos gestionados. Las claves SSH son la forma más común de proporcionar acceso, pero también se admiten otras formas de autenticación.

Mientras que los módulos proporcionan los medios para llevar a cabo una tarea, la forma de utilizarlos es a través de un playbook de Ansible. Un playbook es un archivo de configuración escrito en YAML que proporciona instrucciones sobre lo que hay que hacer para poner un nodo gestionado en el estado deseado. Los playbooks son simples, legibles y autodocumentados. También son idempotentes, lo que significa que un playbook se puede ejecutar en un sistema en cualquier momento sin tener un efecto negativo sobre él. Si un playbook se ejecuta en un sistema que ya está correctamente configurado y en el estado deseado, ese sistema debería seguir estando correctamente configurado después de que se ejecute el playbook.

Un playbook puede ser muy simple, como éste que instala, como usuario privilegiado, el servidor HTTP Apache en cualquier nodo del grupo de servidores web de un departamento de TI:

---

- name: Apache server installed

hosts: webservers

become: yes

Los playbooks también pueden ser muy complejos, con condicionales y variables. Sin embargo, dado que la mayor parte del trabajo real lo realizan los módulos de Ansible, los playbooks siguen siendo breves, legibles y claros aunque puedan orquestar redes enteras de nodos gestionados.

# Desarrollo

Al comenzar este apartado de desarrollo nos centraremos en explicar el punto desde el que se parte al inicio del proyecto, esto es, los recursos con los que se cuentan. Estos recursos son aquellos provistos por la empresa a la hora de desarrollar el proyecto. Se planteará de igual manera una planificación sobre el uso de estos recursos en la que, a partir del análisis del trabajo de cada tipo de nodo en la plataforma, se establecerán los recursos que se destinarán a cada máquina. Esto es con el objetivo de conseguir una plataforma funcional ajustando al máximo los recursos utilizados.

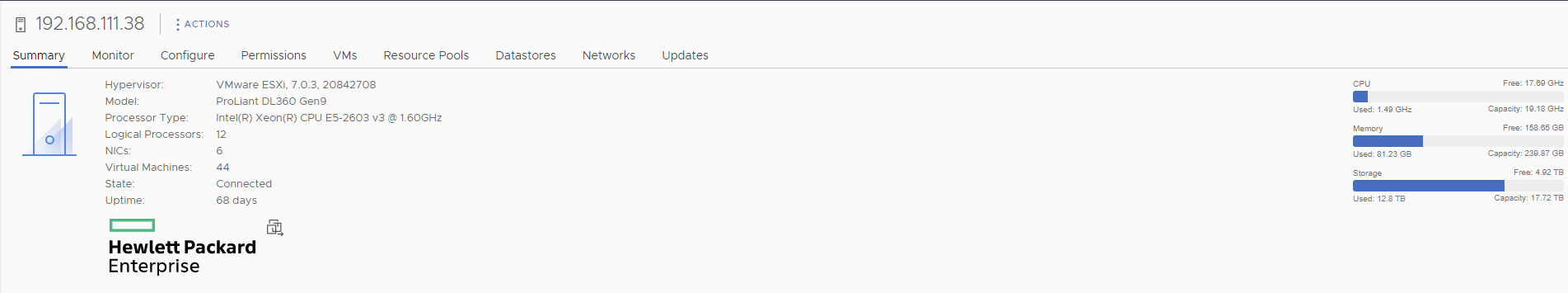
El objetivo de esto es dar una visión general de la topología final del clúster, en la que se incluyan las IPs que se han elegido para cada máquina y cómo serán las comunicaciones dentro de la red.. Por otro lado, con el diseño de alto nivel lo que se busca es mostrar los componentes que forman un clúster de Kubernetes y en que tipo de máquina se aloja cada uno.

Para finalizar el capítulo se propondrá un método de automatización de la instalación con el objetivo de demostrar las posibles ventajas y desventajas respecto al anterior método. Se busca conseguir un método que optimice el tiempo y minimice la configuración manual necesaria en la instalación. El objetivo es un clúster de máquinas virtuales con Kubernetes y Ansible que pueda, al sufrir un desastre, recuperarse prácticamente al completo.

Es importante saber que, además de la capacidad de recuperar las máquinas y la plataforma Kubernetes instalada sobre ellas, la realización periódica de copias de seguridad de diferentes componentes es clave en situaciones de desastre. Por esto, se quiere también desarrollar una estrategia de “backup” con etcd en la que se planeé la realización de copias de seguridad de estos componentes.

### Prerrequisitos y preparación del entorno

El plan inicial es contar con un nodo virtualizado a partir de una máquina física en un centro de datos. Nuestro nodo cuenta con el virtualizador VMware ESXi 7.0.03 que usaremos para crear máquinas virtuales para nuestro clúster. En cuanto a sus características, dispone de un total de 240 GB de memoria principal y una capacidad de almacenamiento de 17.72 TB.

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

### Planificación de uso de recursos (Requisitos de hardware y software)

Los nodos “master” se encargarán de la gestión de la plataforma, la carga de trabajo de los maestros será muy similar en cualquier escenario, ya que los procesos que están funcionando no varían de un caso a otro. Se considera, por tanto, que para este tipo de nodos es suficiente con los recursos que se recomiendan en la documentación de Kubernetes.

Según dicta Kubernetes los nodos “worker” alojarán los aplicativos que se quieran desplegar en la plataforma. Esto determina que aquellos nodos que alojen las aplicaciones han de ser, por norma general, los que necesiten una mayor cantidad de recursos asignados. Se considera en este caso que el número de “workers” será de 2. Se amplían también los recursos destinados a cada nodo a 4 vCPUs, la memoria RAM a 16 GB y un disco de 56GB.

Además de estas máquinas, que son sobre las que se instalará la plataforma, van a ser necesarias otras máquinas como la de bastión, que se utilizará para acceder a la plataforma, la de “services”, que alojará servicios como DNS que servirá para resolver las diferentes direcciones del clúster por nombre, la de DHCP para asignar IPs libres en nuestra red a las máquinas y finalmente la de LDAP para sincronizar a las personas de la organización y sus permisos con la plataforma. ¿¿¿VIPa???

Finalmente, los recursos totales destinados harían un total de:

- vCPU: 1 master (1 máquina \* 4 vCPUs) + 2 workers (2 máquinas \* 4 vCPUs) + 1 registry (1 máquina \* 4 vCPUs) + 2 loadbalancer (2 máquinas \* 4 vCPUs) + 1 bastión (1 máquina \* 4 vCPUs) + 1 VIPa (1 máquina \* 4 vCPUs) = **28 vCPUs**   
- RAM: 1 master (1 máquina \* 16 GB) + 2 workers (2 máquinas \* 16 GB) + 1 registry (1 máquina \* 16 GB) + 2 loadbalancer (2 máquinas \* 16 GB) + 1 bastión (1 máquina \* 16 GB) + 1 VIPa (1 máquina \* 16 GB) = **112 GB**

- Almacenamiento: 1 master (1 máquina \* 56 GB) + 2 workers (2 máquinas \* 56 GB) + 1 registry (1 máquina \* 56 GB) + 2 loadbalancer (2 máquinas \* 56 GB) + 1 bastión (1 máquina \* 56 GB) + 1 VIPa (1 máquina \* 56 GB) = **392 GB**

### 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 |
| 192.168.112.190 | bastion.cin | Kubernetes Bastion |
| 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.

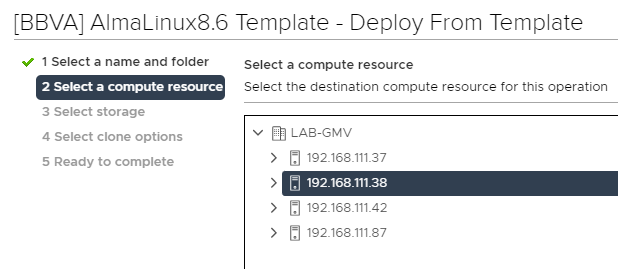
* Red: 192.168.112.0/24
* GW: 192.168.112.254
* DNS: 192.168.131.11
* NTP: 192.168.131.11
* Proxy Navegación: 192.168.131.13:80 | **10.200.140.230:80**

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.

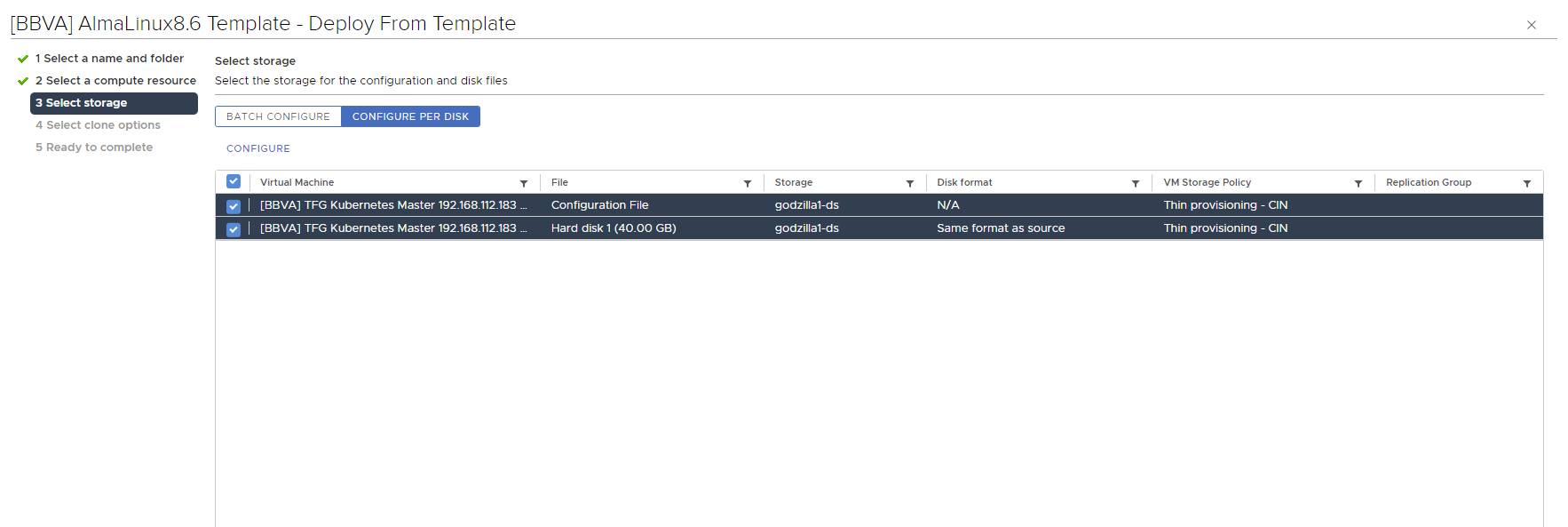
Procedemos a crear la VM del master1 desde la plantilla existente en el laboratorio.

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.

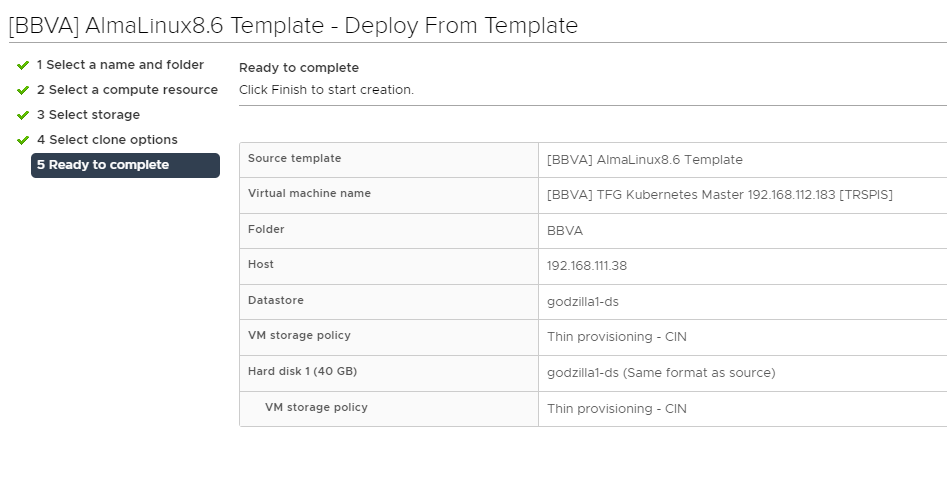
* Para ello pinchamos en el template y marcamos la opción *“New VM from This Template”*
* Seleccionamos el equipo sobre el que queremos desplegarla (192.168.111.38).



* 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

### Instalación y configuración de Ansible

### 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.

### Definición de las tareas y roles necesarios

### Configuración de los hosts y los grupos

### Creación de variables y templates

### Instalación de common necesarios

### 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

### Instalación de los componentes del clúster (master/etcd, workers, bastión, registry, loadbalancer)

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 )

### 

### Configuración de los nodos y los componentes

# Resultados

### Verificación del funcionamiento del clúster (gestionado con Rancher Desktop)

### 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/>

# Conclusiones

### Ventajas de utilizar Ansible para montar un clúster de Kubernetes

# Futuras tendencias y vista a futuro

### Otras formas de instalar en k8s con Helm

# Bibliografía

1. 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.
2. 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>
3. Kubernetes,<https://kubernetes.io/>

**DESCRIPCIÓN DE LAS TAREAS**

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.

