

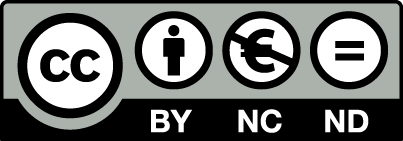
*Infraestructura IT integrada en Kubernetes*

*(SAL\_kubernITes – SAL\_kITs)*

***· Autor***: **S**aúl **A**ltoubah **L**eón

***· Grupo***: ASIA/B

**· *Curso***: 2021 - 2022



Esta obra está sujeta a una licencia de [Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional (CC BY-NC-ND 4.0)](https://creativecommons.org/licenses/by-nc-nd/4.0/deed.es_ES)

Copyright © 2021 **Saúl Altoubah León**

Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

**Centro educativo**

|  |  |  |  |
| --- | --- | --- | --- |
| Código | Centro | Concello | Año académico |
| 15005397 | I.E.S. Fernando Wirtz Suárez | A Coruña | 2021/2022 |

**Ciclo formativo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Código de la familia profesional | Familia profesional | Código de ciclo formativo | Grado | Régimen |
| FP16 | Informática y Comunicaciones | CSIFC01 | Administración de Sistemas Informáticos en Red | Adultos |

**Módulo profesional y unidades formativas de menor duración (\*)**

|  |  |
| --- | --- |
| Código MP/UF | Nombre |
| MP0373 | Proyecto de Administración de Sistemas Informáticos En Red  Equivalencia en créditos ECTS: 5  Código: MP0379  Duración: 26 horas |

**Profesorado responsable**

|  |  |
| --- | --- |
| Tutora | Equipo docente |
| Calvo Castro, Ruth | Aira Martín, Sandra  Corbelle Mejuto, Manuel  Martínez Mejuto, Teresita  Pérez Mariño, Sofía  Romarís Caamaño, Nuria  Vázquez Abelleira, Arturo |

Índice

**· [1. Introducción2](#introducción_01)**

**· [2. Objetivo6](#objetivo_02)**

**· [3. Descripción7](#descripción_03)**

· [3.1. Alcance7](#alcance_03_1)

· [3.2. Medios a utilizar8](#medios_a_utilizar_03_2)

**· [4. Ejecución18](#ejecución_04)**

· [4.1. Diseño20](#diseño_04_1)

· [4.2. Implementación25](#implementación_04_2)

· 4.3. Administración¿?

**· [5. Planificación¿?](#planificación_05)**

· [5.1. Presupuesto¿?](#presupuesto_05_1)

· 5.1.1. Presupuesto¿?

**·** [**6. Bibliografía¿?**](#bibliografía_06)

Índice de figuras

**·** [**Figura 1 – “Servidor de PRIMERGY cumple 25 años”2**](#figura_01)

**·** [**Figura 2 – “Es el fin de la Infraestructura como servicio”3**](#figura_02)

**·** [**Figura 3 – “Dashboard de Netdata”10**](#figura_03)

**·** [**Figura 4 – “GUI de Apache Guacamole”13**](#figura_04)

**·** [**Figura 5 – “Interfaz gráfica de Lens”14**](#figura_05)

**·** [**Figura 6 – “Esquema de conexión VPN de Tailscale”15**](#figura_06)

**· [Figura 7 – “Plano de control personal de Notion”16](#figura_07)**

**· [Figura 8 – “Plano de control personal de GitHub”17](#figura_08)**

**· [Figura 9 – “Esquema resumido de los dispositivos físicos en la infraestructura”20](#figura_09)**

**·** [**Figura 10 – “Arquitectura de Kubernetes”21**](#figura_10)

**·** [**Figura 11 – “Arquitectura de Minikube”22**](#figura_11)

**·** [**Figura 12 – “Esquema general de contenedores con sus respectivas herramientas”23**](#figura_12)

Índice de tablas

**·** [**Tabla 1 – Requerimientos mínimos para VM servidor SAL\_KubernITes24**](#tabla_01)

**·** [**Tabla 2 – Requerimientos mínimos para VM cliente SAL\_kITs24**](#tabla_02)

**·** [**Tabla 3 – Lista de comandos para la instalación de CLI de Docker25**](#tabla_03)

**·** [**Tabla 4 – Lista de comandos para la instalación del clúster de Minikube26**](#tabla_04)

**· Tabla 5(pendiente)**

[…]

**· 1. Introducción**

Hace una década, ejecutar máquinas virtuales (***VM,*** ***Virtual Machines***) en la nube era algo de vanguardia. Esto hizo que la migración a la nube fuera bastante simple: las empresas solo podían cambiar las VM que ya ejecutaban en los servidores de sus instalaciones a los servidores de un proveedor de infraestructura como servicio (***IaaS, Infrastructure As A Service***). Las empresas, al librearse de la carga del mantenimiento de los servidores físicos, ganaron flexibilidad y redujeron los costos, pero en la actualidad, nadie desarrolla nuevas aplicaciones basadas en las VMs. En cambio, recurren a un modelo que es más rentable, exige menos mantenimiento y es más escalable de lo que las VMs podían esperar ser: **la contenerización**. Este modelo, no las VMs, representan el futuro de la computación.

Los antecedentes recientes de la infraestructura de TI son la historia de abstracción. En los 90’s, se ejecutaban las aplicaciones en hardware en bastidores físicos:

**Figura 1 - Servidor PRIMERGY de Fujitsu cumple 25 años**

Cada nuevo desarrollo ha abstraído cada vez más aplicaciones del hardware. Por este motivo, la parte de infraestructura que las empresas tenían que manejar se redujo muchísimo, pero el modelo ***IaaS*** de ejecutar las VMs en la nube no es el último paso en esa progresión. Las VMs tienen algunas desventajas significativas:

* El hecho de que cada VM ejecute sistemas operativos múltiples inevitablemente crea ineficiencias. Incluso cuando se escala y dimensionan de manera adecuada, lo que no es seguro, las VM dejan mucha capacidad sin usar en los servidores.
* Con las VMs, las empresas siguen siendo responsables de llevar a cabo los molestos ejercicios de operaciones, como la recuperación de desastres, la alta disponibilidad y el escalado, así como la aplicación de parches y la seguridad.
* Las VMs no son muy flexibles y funcionan de manera diferente en distintos hiperescaladores, por lo que una VM impulsada por ***Microsoft Azure*** no se puede migrar a ***AWS*** ni a ***Google Cloud***.



**Figura 2 - Es el fin de la infraestructura como servicio**

Las empresas que todavía tratan de migrar a la nube mediante migración de sus VM deberían pensárselo dos veces. Comprometerse ahora con un modelo ineficiente retrasará el progreso en el futuro. En cambio, las empresas deberían recurrir a la contenerización, aunque deban hacerse cambios significativos en sus procesos. Los contenedores son el próximo paso en la tendencia de la abstracción. Varios contenedores pueden ejecutarse en un solo núcleo del sistema operativo, lo que significa que usan recursos de manera más eficiente que las VM. De hecho, en la infraestructura necesaria para una VM, podría ejecutarse una docena de contenedores. Sin embargo, los contenedores tienen sus desventajas. Si bien tienen más espacio eficiente que las VM, ocupan capacidad de infraestructura cuando están inactivos y generan costos innecesarios.

La transición a la contenerización requiere cambios importantes en los procesos y en la estructura de sus equipos de TI, así como elecciones justificadas sobre cómo llevarla a cabo.

El traspaso a la infraestructura de TI moderna es tanto una transformación de personas y procesos como de tecnología. La administración de la infraestructura de TI tradicional se basa, en gran medida, en las soluciones manuales de apuntar y hacer clic. Por el contrario, la administración de la infraestructura contenerizada es más como la ingeniería del software: los equipos de TI usan un código para describir el resultado final que quieren y sistemas automáticos para llevar a cabo la implementación.

Para aprovechar al máximo la flexibilidad y la eficacia que logra la infraestructura moderna, los equipos de TI deben pasar a lo que se conoce como una orientación de ***DevOps***, que aporta prácticas de desarrollo de software ágiles a la administración de infraestructura. Por ejemplo, los equipos de TI de las empresas tradicionales tienden a estar aislador por función, pero ***DevOps*** toma un enfoque más ágil donde un equipo posee la totalidad de la aplicación. Es necesario adaptar esta nueva manera de trabajar para lograr el éxito de la transición de su infraestructura.

Es conveniente evitar soluciones de terceros de propiedad privada ya que muchas pilas de software de este estilo afirman que facilitan la transición a los contenedores. Sin embargo, al final, estas “abstracciones” solo pueden agregar pasos y costos adicionales; si bien pueden simplificar la transición inicial, a medida que se perfecciona, con el tiempo, tendrá necesidades que, probablemente, no puedan manejar. En cambio, nosotros deberíamos eliminar al intermediario desde el principio y utilizar las soluciones de ***open source*** (código abierto) con las comunidades activas, como ***Docker*** y ***Kubernetes***. Es posible que la curva de aprendizaje sea más pronunciada, pero nos ahorrará tiempo a nuestros equipos de TI en un futuro.

No se tiene que hacer el traslado a los contenedores de una sola vez. Eso sería sumamente complejo. En cambio, podemos trasladar algunos servicios a los contenedores poco a poco hasta estar completamente contenerizados. Esta estrategia de adopción progresiva se conoce como “método estrangulador”, debido a que el código nuevo lentamente “estrangula” o reemplaza al código viejo. Esta estrategia lenta y constante también les da a nuestros equipos de TI tiempo para adaptarse a las nuevas maneras de trabajar.

Y en relación a este proyecto, ya que se está utilizando esta tecnología, ¿se podría decir que es el comienzo de un nuevo concepto de servicio cuyo nombre sería ***CIaaS (Containered Infrastructure As A Service)***?

**· 2. Objetivo**

El presente documento contiene una guía para ejecutar cumplir con objetivo principal de este proyecto se basará en el diseño, implementación y administración de una infraestructura general de TI que abarcará todos los requerimientos necesarios para su utilización: **monitorización**, **seguridad**, **almacenamiento de datos**, **acceso remoto**, etc. Integrado en el sistema de contenedores mediante ***Docker***, orquestado mediante ***Kubernetes*** y herramientas variadas para hacerlo posible y creíble para su creación.

Dicho en otras palabras, desarrollará una idea/concepto general de una infraestructura IT contenerizada o de contenerización que abarcan sus respectivas características conocidas, solo que, dotándola de:

· 1. **Flexibilidad**, para poder estar conectado desde cualquier equipo sea donde sea.

· 2. **Escalabilidad**, para una respuesta rápida sin encontrarse con inconveniencias como la saturación de procesos ya que los contenedores son ligeros y no se sobrecargan ya que son controlados por un único equipo.

· 3. **Alta disponibilidad**, para estar totalmente disponible sea el momento que sea.

· 4. **Compatibilidad**, para que se pueda implementar e implantar en diferentes arquitecturas (local o en la nube y, si se puede el caso, hiperconvergente).

La configuración deberá realizarse en diferentes máquinas con el sistema operativo recién instalado, si bien también se deben llevar a cabo periódicamente sobre cualquier máquina para comprobar todos y cada uno de los estados en el que se encuentran los contenedores implementados.

**· 3. Descripción**

Como breve descripción para este proyecto en relación a lo que se va a tratar, hay que tener en cuenta los siguientes rasgos o requisitos:

**·** Al ser una **infraestructura de tecnologías de la información o TI**, cabe recordar indicar **de qué o cómo está compuesta**.

**· De qué manera se ha llevado a cabo.**

**· Explicar y describir qué medios/recursos se han empleado para su creación.**

**· Etc.**

**· 3.1. Alcance**

Este documento se ha elaborado para proporcionar información específica con objeto de implementar tecnologías de contenerización sobre una empresa, o incluso con la oportunidad de ser de carácter particular, con el sistema operativo ***Linux***, instalado, tanto en inglés como en español en su última versión.

El documento incluye:

* Herramientas que fueron utilizadas para la creación de dicho proyecto y hacerlo posible.
* Ejecución de las fases de diseño de la infraestructura acompañadas de esquemas y capturas de pantalla para idear una base del concepto de “*Infraestructura de contenerización o contenerizada*”.
* Tablas de presupuesto para realizar una valoración económica de la misma.

**· 3.2. Medios a utilizar**

Para el desarrollo de este proyecto serán necesarias diversas herramientas y más adelante se describirá los medios a utilizar. Estas serán las herramientas a utilizar:

***VMware***

***VMware*** es un software de virtualización que ofrece la posibilidad de virtualizar diversos sistemas operativos para todo tipo de máquinas, tanto ordenadores personales como servidores de empresas.

Gracias a ***VMware*** dispondremos de un entorno de implementación de sistemas aislado. Especialmente, usaremos su nueva versión ***Workstation Pro 16*** que mejora la integridad gráfica y el rendimiento de las VMs en las cuales vamos a crear y diseñar nuestra infraestructura.

**Sistemas Operativos**:

* Linux:
  + ***Ubuntu 20.04 Server***, también conocido como “***Focal Fossa***” lanzado el 20 de abril de 2020, es una distribución de Linux cuya implementación está orientada a servidores. Es un sistema de código abierto, este sistema operativo será el indicado para nuestra infraestructura.
  + ***Ubuntu 20.04 Desktop***, mismas características que su versión ***Server***, pero es utilizado a nivel de equipos clientes.
  + ***Linux Mint 20***, conocido como “’***Ulyana***’ ***XFCE Edition***” basada en ***Ubuntu 20.04***, es una de los sistemas operativos alternativos que podemos implementar en nuestra infraestructura tanto a nivel de servidor y/o a nivel de cliente. Se caracteriza por ser liviano, veloz y por disponer un escritorio gráfico llamado “***Cinnamon***” para aquellos usuarios familiarizados con el escritorio de ***Windows***.

***\*****Se había empleado como sistema operativo de pruebas antes de implementarlo en el proyecto para comprobaciones de rendimiento y validez para nuestra infraestructura contenerizada.*

***Docker***

***Docker*** es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización a nivel de sistema operativo en ***Linux***. Esto evita la sobrecarga de realizar máquinas virtuales únicamente dedicadas a una aplicación.

Es sin duda una de las herramientas principales para el desarrollo del proyecto, puesto que es esencial para la contenerización de aplicaciones. Su gran comunidad de usuarios hace que sea un entorno de desarrollo muy amplio y con múltiples posibilidades.

***Kubernetes***

***Kubernetes*** (***k8s***) es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios. Google liberó el proyecto Kubernetes en el año 2014 y se basa en la ejecución de aplicaciones en producción a gran escala junto a las mejores ideas y prácticas de la comunidad.

Tiene varias características, se puede pensar en ***Kubernetes*** como:

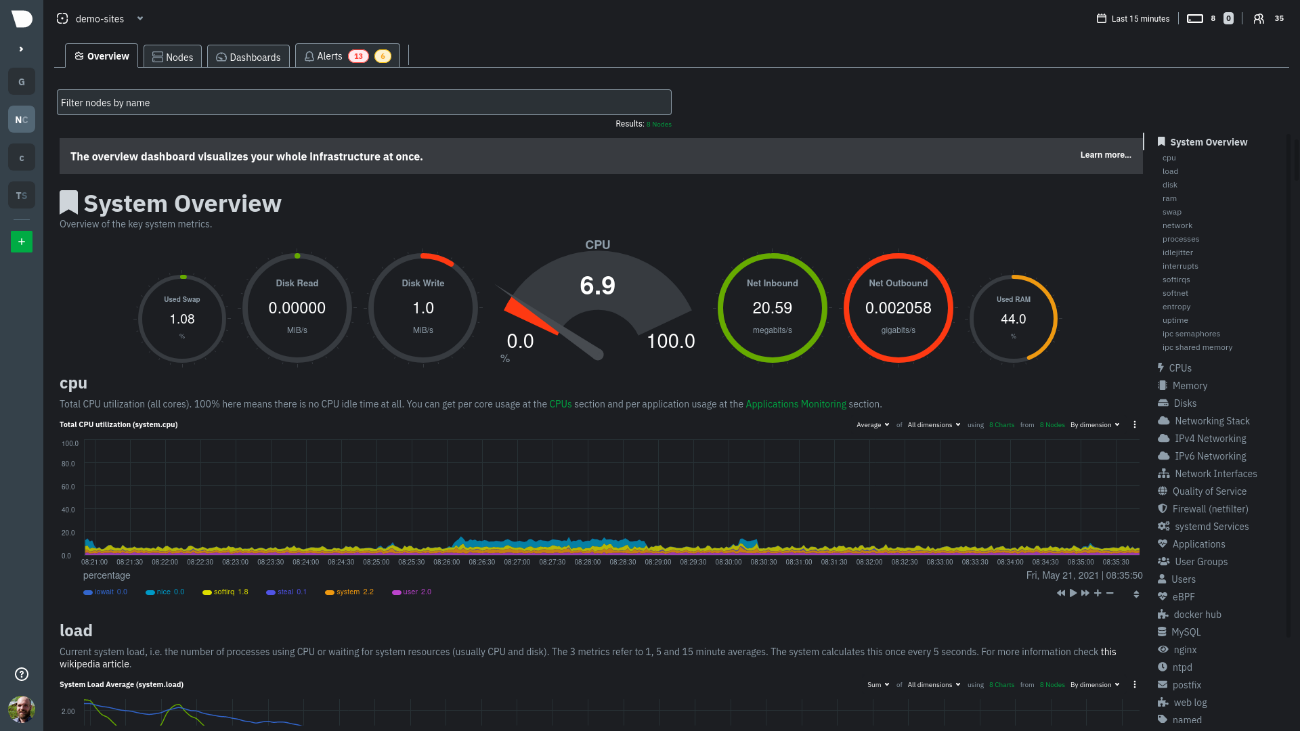
* Una plataforma de contenedores
* Una plataforma de microservicios
* Una plataforma portable de la nube

Y mucho más…

***Kubernetes*** ofrece un entorno de administración centrado en contenedores. Orquesta la infraestructura de cómputo, redes y almacenamiento para que las cargas de trabajo de los usuarios no tengan que hacerlo. Opera a nivel de contenedor y no a nivel de hardware, ofrece algunas características que las ***PaaS*** (***Platforms As A Services***) también ofrecen, como ***deployments*** (despliegues), escalado, balanceo de cargas (***load balancing***), registros (***logs***) y monitoreo.

***Netdata***

***Netdata*** es una herramienta de monitorización que soluciona problemas de ralentización y anomalías en nuestra infraestructura con miles de métricas por segundo, visualizaciones significativas y unas perspicaces alarmas de salud (refiriéndose a el estado en que se encuentran de los procesos) sin ninguna configuración necesaria.



**Figura 3- Dashboard de Netdata**

Es gratuíto, sus agentes de monitorización trabajan con Netdata Cloud para ayudarnos a monitorizar y solucionar problemas en cada capa de nuestros sistemas para encontrar debilidades antes de que se vuelvan en cortes de servicios.

***Helm***

***Helm*** es una herramienta para gestionar aplicaciones de ***Kubernetes***. ***Helm*** nos ayuda a “timonear” ***Kubernetes*** usando “cartas de navegación” (***charts***), conocidas como ***Helm Charts***. La principal función de Helm es definir, instalar y actualizar aplicaciones complejas de Kubernetes. Helm es mantenido por la ***CNCF, Cloud Native Computing Fundation*** (Fundación de Informática Nativa en la Nube) en colaboración con ***Microsoft***, ***Google***, ***Bitnami*** y la comunidad de ***Helm***.

Con ***Helm Charts*** es posible crear, versionar y publicar una aplicación ***Kubernetes***. Cuando usamos los charts tenemos un asistente de optimización que facilita la administración e instalación de las aplicaciones ***Kubernetes*** y el proceso de empaquetamiento.

***Nginx***

***Nginx*** es un servidor web de código abierto que, desde su éxito inicial como servidor web ahora también es usado como proxy inverso, cache de HTTP y ***load balancer***. Está diseñado para ofrecer un bajo uso de memoria y alta concurrencia (procesamiento de múltiples solicitudes que se ejecutan al mismo tiempo). En lugar de crear nuevos procesos para cada solicitud web, ***Nginx*** usa un enfoque asincrónico basado en eventos donde las a solicitudes se manejan en un solo hilo. Un proceso maestro puede controlar múltiples procesos de trabajo. El proceso maestro mantiene los procesos de trabajo, y son estos lo que hacen el procesamiento real. Algunas características comunes que se ven en Nginx incluyen:

* Proxy inverso con caché (servidor intermediario que se asienta entre el cliente y el servidor maestro para el envío de tráfico directamente a los clientes)
* IPv6
* Balanceo de carga (***load balancing***)
* ***Websockets*** (para establecer una conexión bidireccional entre ***frontend*** (interfaz) y ***backend*** (servidor)
* Certificación ***TLS/SSL***

***Let’s Encrypt***

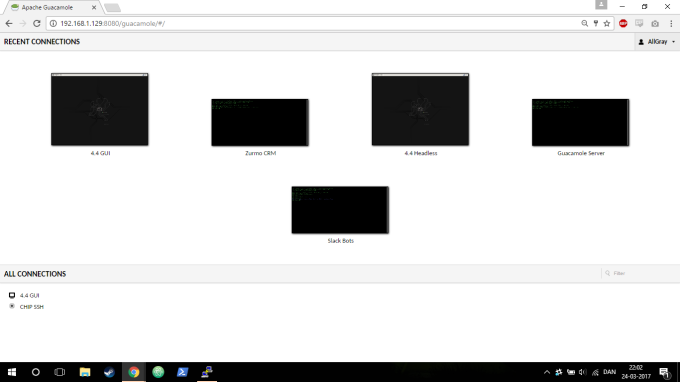
***Let’s Encrypt*** es una autoridad de certificación (***CA, Certification Authority***) gratuíta, automatizada y abierta para el beneficio del público. Es un servicio provisto por el ***Internet Security Research Group (ISRG)***. Distribuyen certificados digitales gratuitamente a personas que necesitan poder habilitar el uso del protocolo ***HTTPS (SSL/TLS)*** en sitios web asegurando la privacidad y la seguridad a los usuarios. Básicamente, su objetivo es hacer posible la configuración de un servidor ***HTTPS*** y hacer que obtenga automáticamente un certificado confiado por el navegador, sin ninguna intervención humana. Esto se logra ejecutando un agente de manejamiento de certificados en un servidor web.

***Cloudflare***

***Cloudflare*** es una de las redes más grandes del mundo. Hoy en día, los sitios web y las aplicaciones de las empresas, organizaciones, etc. con presencia de Internet son más rápidos y seguros gracias a ***Cloudflare***. Es una empresa que proporciona una red de entrega de contenido, servicios de seguridad de Internet y servicios de servidores de nombres de dominio distribuidos, localizados entre el visitante y el proveedor de alojamiento del usuario de Cloudflare, y que actúan como proxy inverso para sitios web.

***Apache GuacamoleTM***

***Apache Guacamole*** es una puerta de enlace de acceso remoto sin cliente (***clientless***). Soporta protocolos estándar como ***VNC*** (Linux), ***RDP*** (Windows) y ***SSH***. Se le llama ***clientless*** porque no necesita de ningún plugin o de software cliente. Gracias a ***HTML5***, una vez que Guacamole esté instalado en un servidor, todo lo necesario para acceder a nuestros escritorios es a través de un buscador web.

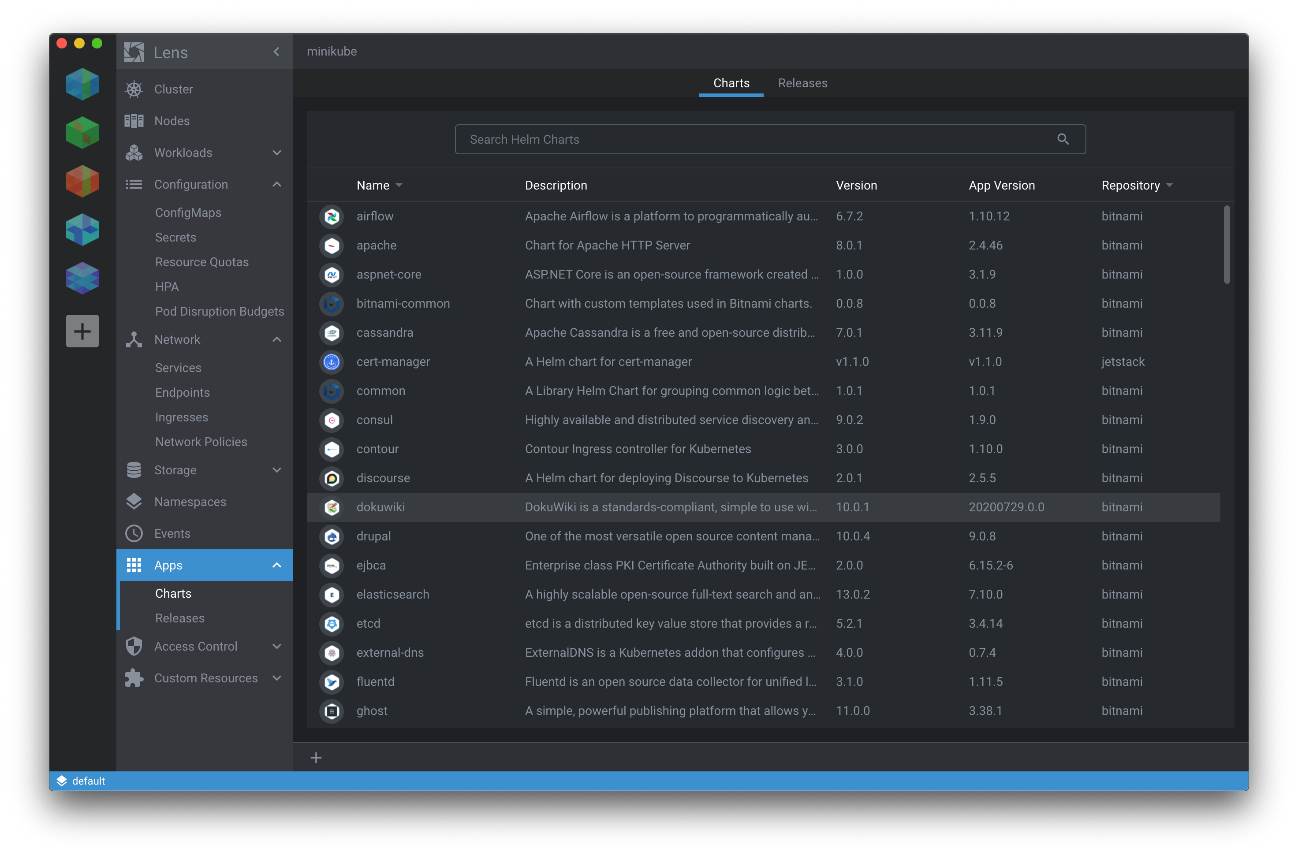


**Figura 4 - GUI de Apache Guacamole**

***Lens***

***Lens*** es un IDE de código abierto para administrar clústeres de Kubernetes multiplataforma. A través de ***Lens***, podemos administrar fácilmente varios clústeres de ***Kubernetes***. Algunos beneficios de usar **Lens** incluyen:

* Confianza en que nuestros clústeres están configurados correctamente.
* Mayor visibilidad, estadísticas en tiempo real, flujos de registros y capacidades prácticas de resolución de problemas (si disponemos de ***Grafana***).
* La capacidad de trabajar con nuestros clústeres de forma rápida y sencilla, mejorando radicalmente la productividad y la velocidad del negocio.

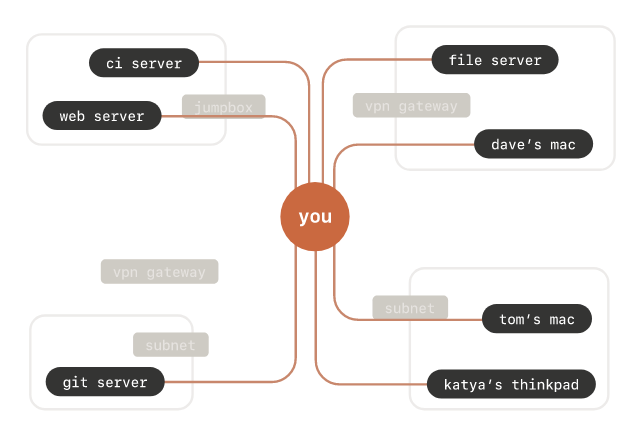


**Figura 5 - Interfaz gráfica de Lens**

***Freenom***

***Freenom*** es un proveedor de dominios públicos gratuitos del mundo, facilita la construcción de sitios web y sus contenidos, para cualquier empresa o persona de forma gratuita. Emplea un método de direccionamiento y enrutamiento llamado ***AnyCast*** en el que las peticiones entrantes pueden ser encaminadas a una variedad de diferentes localizaciones o nodos. ***Freenom*** garantiza la estabilidad y el rendimiento de todos los dominios que administra.

***Tailscale***

***Tailscale*** es una VPN que crea una red segura entre nuestros servidores, equipos e instancias en la nube; incluso cuando está separada por firewalls o subredes. A pesar de otras VPNs, configurando una nueva red en ***Tailscale*** requiere de una mínima configuración a nivel de usuario y sin experiencia con redes. Los dispositivos se conectan directamente, trabajando desde cualquier localización física o entorno de red.

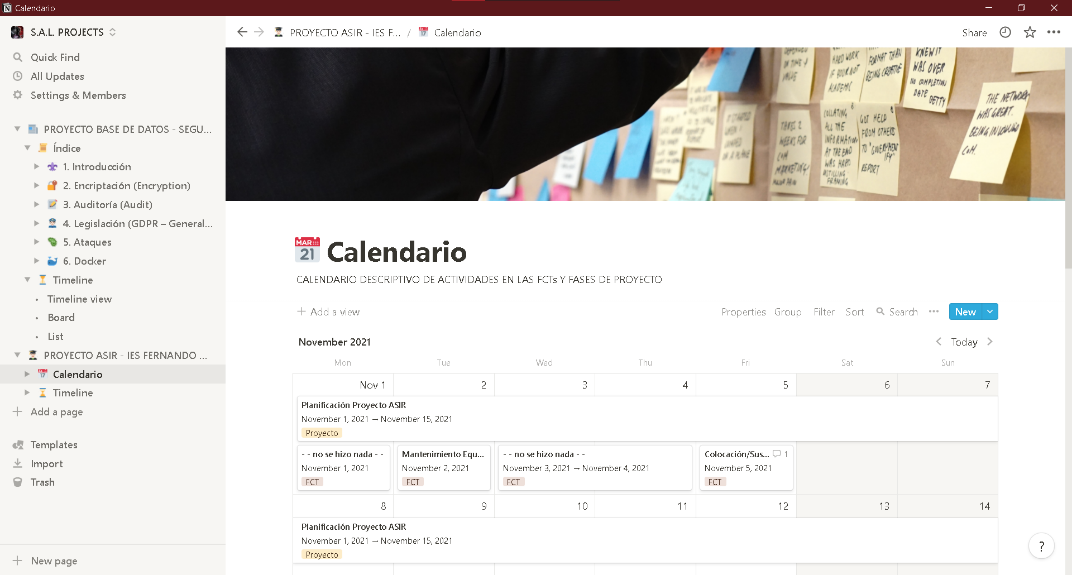
**Figura 6 - Esquema de conexión VPN de Tailscale**

***Notion***

***Notion*** es un sistema de organización de información. Aglutina en una sola app las funciones de otras muchas que usamos frecuentemente, como editor de textos, toma de notas, gestión de tareas y proyectos, etc. Desde un punto de vista práctico, se puede decir que es una “navaja suiza de la productividad”. Nos permite almacenar de forma ordenada u práctica casi cualquier unidad de información. ***Notion*** permite guardar dos tipos de estructuras de información básicamente:

* Páginas, un documento de texto como si estuviésemos usando un procesador de texto como ***Word***, notas como ***Evernote***, lista de tareas, recordatorios en un calendario, vídeos de ***YouTube***…

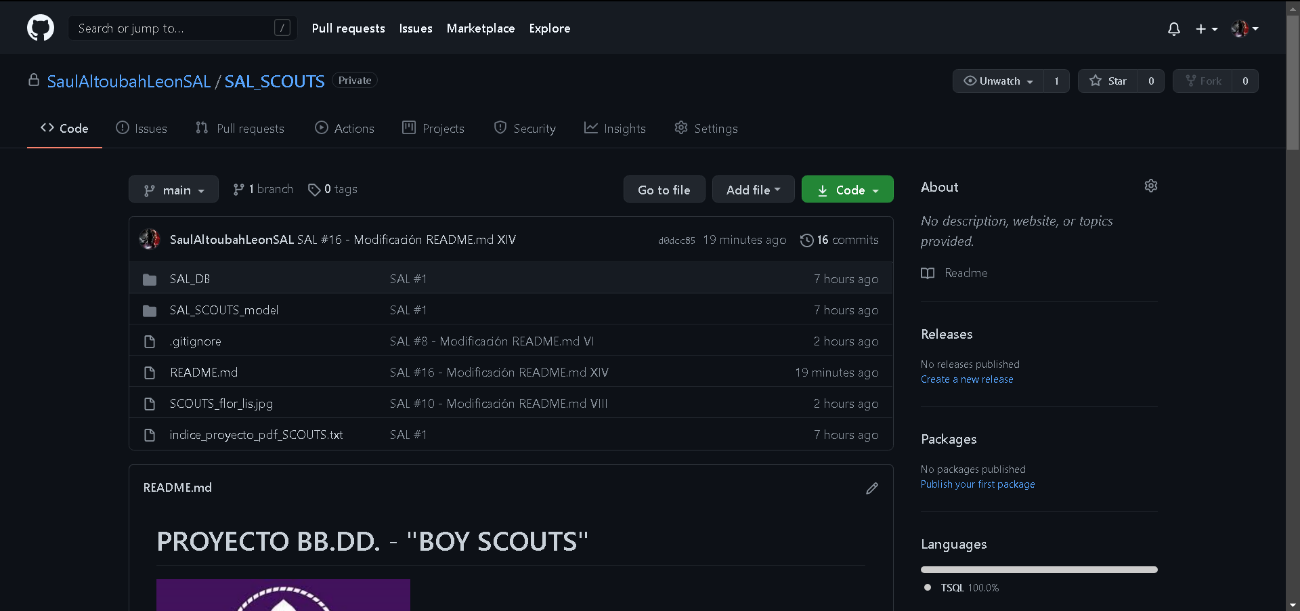
Podemos personalizar cada hoja con una portada superior y un icono si lo deseamos. Además, podremos organizarlo de forma que podremos insertar o enlazar a otras páginas desde cualquier punto de la página, ordenar el contenido en columnas, cambiar el tipo de tipografía, establecer el ancho del documento, etc.

* Bases de datos, poder crear bases de datos totalmente flexibles, sobre la marcha, que nos permiten crear elementos dentro de ellas y a cada elemento asignarle unas propiedades.

**Figura 7 - Plano de control personal de Notion**

***GitHub***

***GitHub*** es un servicio de hosting de repositorios almacenados en la nube. Esencialmente, hace que sea más fácil para individuos y equipos usar ***Git*** como la versión de control y colaboración. La interfaz de GitHub es sencilla de usar, algunas personas lo usan para administrar cualquier tipo de proyectos como desarrollo de código, escribir libros…



**Figura 8 - Plano de control personal de GitHub**

Estas serán las herramientas utilizadas y alguna que otra que no se haya nombrado que posiblemente se le alude más adelante a lo largo de este documento. Para más información, avanzar sobre el siguiente punto [· 4. Ejecución](#ejecución_04) en donde se explicarán con más detalle los recursos a emplear.

**· 4. Ejecución**

Para diseñar una infraestructura de tecnologías de la información contenerizada hay que saber primero qué es.

Una infraestructura de tecnologías de la información o TI se trata de un conglomerado de elementos necesarios para operar y gestionar entornos de tecnologías de la información en empresas. Estos elementos incluyen el hardware, el software, los elementos de red, sistemas operativos y almacenamientos de datos. Todos ellos para ofrecer servicios y soluciones TI. Sus elementos se pueden clasificar de la siguiente manera:

* Hardware:
  + Servidores
  + Datacenters (centros de datos)
  + Equipos informáticos
  + Routers
  + Hubs/Switches
  + Etc.
* Software:
  + ***CMS, Content Management Systems*** (Sistemas Gestores de Contenido): ***Wordpress***, ***Moodle***…
  + Sistemas Operativos: ***Windows***, ***Linux*** (Ubuntu, Debian…), ***MacOS***.
  + ***ERP, Enterprise Resource Planning*** (Planificación de Recursos Empresariales): ***Odoo***, ***Holded***, ***Selenne***…
  + Etc.

Puede implementarse en un sistema de ***cloud computing*** (informática en la nube) o en instalaciones tradicionales:

* Tradicional: también conocida como ***on-premise*** (local), las empresas son las propietarias de todos los elementos (servidores, almacenamiento de datos, entre otros), a los cuales gestionan en sus propias instalaciones
* En la nube (***Cloud Computing***): la infraestructura en la nube hace referencia a los elementos y los recursos que se necesitan para el ***cloud computing***. Existen 3 tipos: pública, privada e híbrida.
* Hiperconvergente: permite gestionar los recursos informáticos, de red y de almacenamiento desde una sola interfaz. Así podrá admitir cargas de trabajo más modernas con arquitecturas escalables en el sector a través de la combinación del almacenamiento de datos y la informática definidos por software.

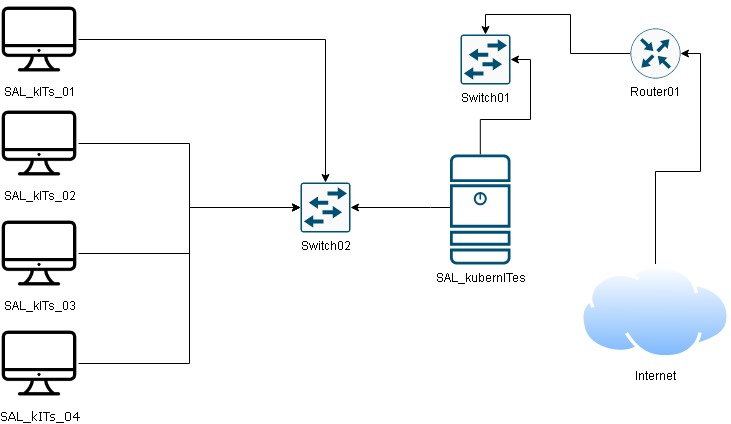
Y ahora, en relación a todo lo comentado en el punto [· 1. Introducción](#introducción_01), ¿cómo podemos definir una infraestructura IT contenerizada (“*integrada en Kubernetes*”)?

Podríamos decir que, se trataría de dicha infraestructura optimizada en escalabilidad y flexibilidad gracias al uso de contenedores para agilizar su funcionalidad a la hora de ofrecer servicios y soluciones TI que se adapta a cualquier implementación que se quiera realizar, tanto si se instala en servidores ***bare-metal***, en nubes públicas, privadas o híbridas, o se preconfigura y se implanta en servidores dedicados hiperconvergentes; de esta manera dotándola de adaptabilidad a cualquier configuración que se nos proponga dependiendo de: *¿a quién se va a dirigir la empresa? ¿cuáles son sus objetivos a alcanzar? ¿y con qué estrategia se van a basar sus acciones de cara a sus clientes?*

**· 4.1. Diseño**

**· Esquema físico**:

En el esquema siguiente se muestra una nueva propuesta de infraestructura diseñada para este proyecto y se detalla cada uno de los componentes.

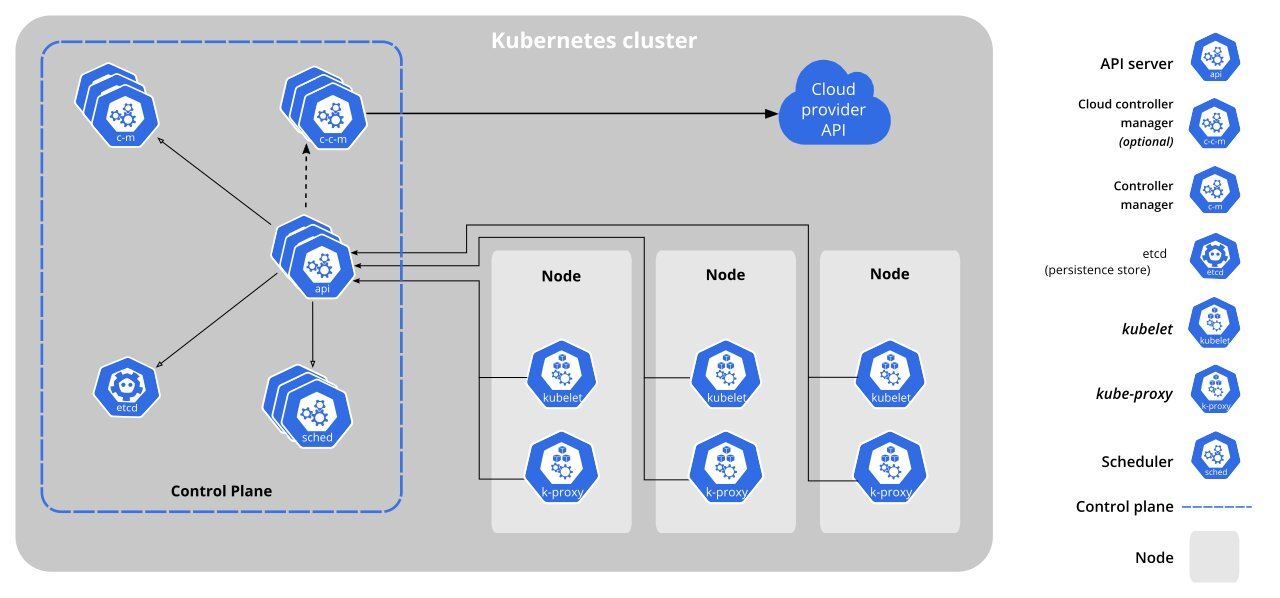


**Figura 9 – Esquema resumido de los dispositivos físicos en la infraestructura**

Como se puede observar en la figura, la infraestructura propuesta para este proyecto se trata de un ejemplo simple que muestra el conexionado de los equipos físicos incluyendo el enrutamiento al que deben seguir.

**· Esquema de contenerización**:

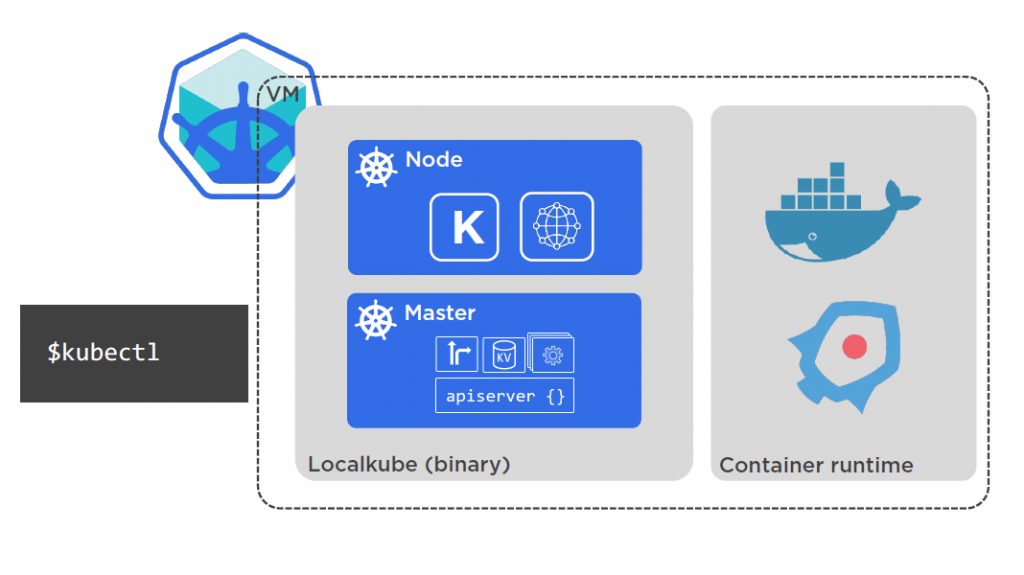
En el esquema siguiente se muestra la arquitectura de Kubernetes recomendada que se iría a implementar en el equipo servidor:



**Figura 10 - Arquitectura de Kubernetes**

Esta arquitectura está formada por los siguientes componentes:

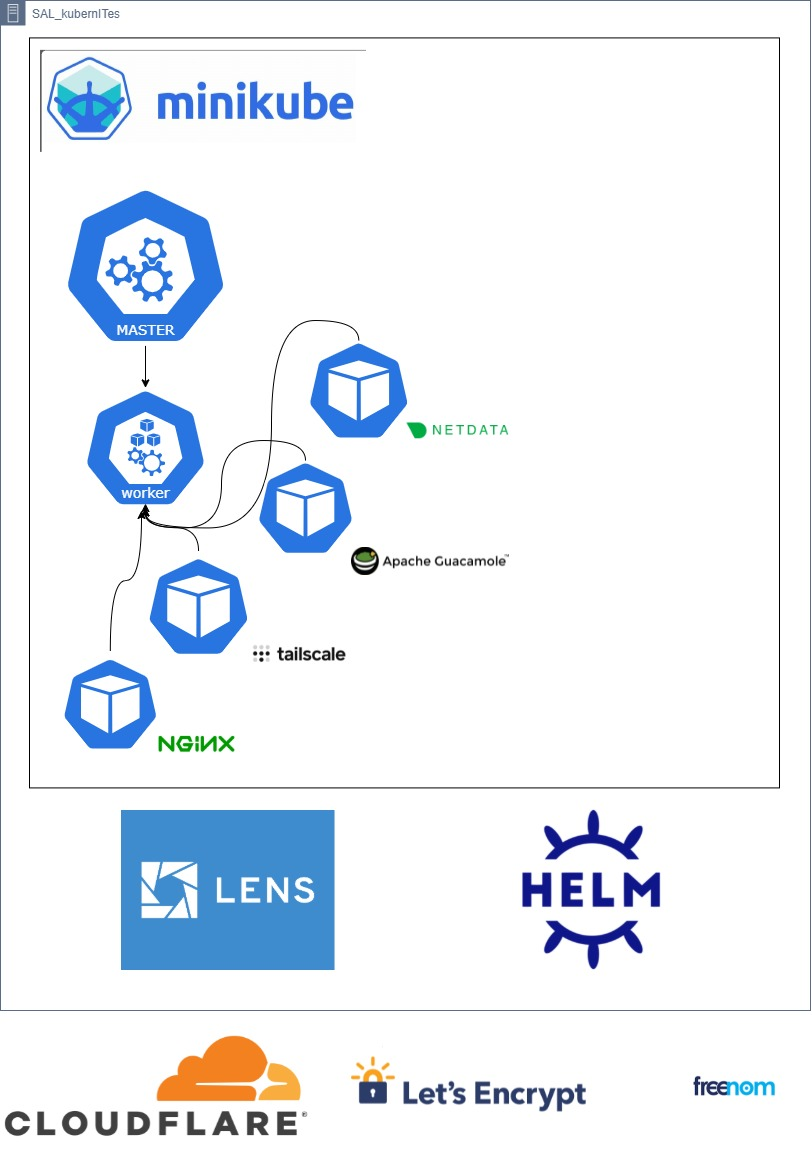
* ***Kube-apiserver***: es el componente del plano de control de ***Kubernetes*** que expone la API de ***Kubernetes***. El servidor API es el ***frontend*** para el plano de control de ***Kubernetes***.
* ***Etcd***: es “la base de datos” de Kubernetes usado como almacenamiento de todas las claves, componentes y datos de todo nuestro clúster.
* ***Kube-scheduler***: es el componente “programador” del plano de control que vigila a los ***Pods*** sin nodo asignado y les asigna uno para ponerlos en funcionamiento.
* ***Kube-controller-manager***: es el componente del plano de control que ejecuta los procesos del controlador. Lógicamente, cada controlador está en un proceso separado, pero para reducir su complejidad, están todos compilados en un único binario y un único proceso.
* ***Kubelet***: un agente que arranca cada nodo en el clúster. Se asegura que los contenedores se están ejecutando en un ***Pod***.
* ***Kube-proxy***: es el proxy de red que se ejecuta en cada nodo en nuestro clúster, implementando parte del concepto de los ***Servicios*** de ***Kubernetes***.
* ***Pod***: es el objeto más pequeño y básico desplegable en ***Kubernetes***, es una unidad básica de ejecución que representa procesos corriendo en nuestro clúster. Tiene la función de encapsular un contenedor (o en algunos casos varios contenedores), recursos de almacenamiento, una dirección IP única que es válida solo dentro de nuestro clúster y las opciones que definen como el contenedor debe ejecutarse.
* ***ReplicaSet***: es una regla definida mediante un ***Deployment*** en ***Kubernetes*** y usado para garantizar la disponibilidad de un número específico de ***Pods*** idénticos.
* ***Deployment***: es un conjunto de órdenes declaradas que se le dan a nuestro clúster de ***Kubernetes*** todos los ***ReplicaSets*** y los ***Pods*** se alinean a lo que diga un ***Deployment***.

Estos son los componentes más conocidos de ***Kubernetes***, pero en este caso vamos a crear una versión simplificada y ligera utilizando ***Minikube***. ***Minikube*** utiliza en un solo clúster el nodo master y el nodo *worker*:

**Figura 11 - Arquitectura de Minikube**

Como podemos observar, este clúster de pruebas se va a ejecutar dentro de una VM en la que se le instalarán los paquetes de instalación de ***Docker*** y de ***Minikube***.

Y finalmente esta será el esquema general de contenedores a seguir para la infraestructura:



**Figura 12 - Esquema general de contenedores con sus respectivas herramientas**

Los requisitos de las VMs que se han seguido para la elaboración del proyecto son los siguientes:

|  |
| --- |
| Servidor SAL\_kubernITes |
| * Almacenamiento interno | **200 GB** |
| * Memoria RAM | **2,5 GB (recomendables 4GB)** |
| * Nº CPUs | **2** |
| * Tarjeta de red | **NAT** |

**Tabla 1 - Requerimientos mínimos para VM servidor SAL\_kubernITes**

|  |
| --- |
| Cliente SAL\_kITs |
| * Almacenamiento interno | **100 GB** |
| * Memoria RAM | **1 GB (recomendables 2GB)** |
| * Nº CPUs | **2** |
| * Tarjeta de red | **NAT** |

**Tabla 2 - Requerimientos mínimos para VM cliente SAL\_kITs**

**· 4.2.Implementación**

**· Sistema operativo del servidor y clientes**: ***SAL\_kubernITes & SAL\_kITs***

El entorno utilizado para el desarrollo es una pequeña máquina virtual con un sistema operativo ***Ubuntu 20.04 Server*** contenido en ella que, a su vez, se ha instalado la versión Desktop en los clientes. Puede o no disponer de interfaz gráfica en caso de querer acceder a por CLI mediante SSH, pero le dotaremos de escritorio gráfico instalándole el escritorio “***Cinnamon***” o el escritorio por defecto que suele venir implantado en su versión ***Desktop***. Si se quiere usar otra distribución de ***Linux*** como ***Linux Mint 20*** como se ha comentado anteriormente, cumple con su propósito igualmente.

**· Instalación de Docker**:

Para la instalación de la CLI de ***Docker***, se deben ejecutar la siguiente lista de comandos:

|  |
| --- |
| Comandos |
| 1. *sudo apt update && sudo apt full-upgrade -y* 2. *sudo apt install apt-transport-https ca-certificates curl software-properties-common -y* 3. *curl -fsSL* [*https://download.docker.com/linux/ubuntu/gpg*](https://download.docker.com/linux/ubuntu/gpg) *| sudo add-key add –* 4. *sudo add-apt-repository “deb [arch=amd64]* [*https://download.docker.com/linux/ubuntu*](https://download.docker.com/linux/ubuntu) *focal stable”* 5. *apt-cache policy docker-ce* 6. *sudo apt install docker-ce -y* 7. *sudo usermod -aG docker ${USER}* 8. *su - ${USER}* 9. *reboot* 10. *id -nG* 11. *sudo service docker status* |

**Tabla 3 - Lista de comandos para la instalación de CLI de Docker**

· ***Instalación de Minikube***:

El siguiente paso es instalar el clúster de ***Minikube***. No es necesario configurar nada ya que solamente es activar el clúster y ya se puede trabajar con él. Los comandos a ejecutar son:

|  |
| --- |
| Comandos |
| 1. *curl -LO* [*https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64*](https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64) 2. *sudo install minikube-linux-amd64 /usr/local/bin/minikube* 3. *minikube start* |

**Tabla 4 - Lista de comandos para la instalación del clúster de Minikube**

Se recomienda ejecutar estos comandos como administrador, no como ***root***. Alternativamente, ***Minikube*** puede descargar la versión de ***kubectl*** y deberíamos ser capaces de usarlo tal que así: ***minikube kubectl – get pod -A***

Para agilizar a la hora de ejecutar el comando ***kubectl*** se recomienda crear un alias: ***alias kubectl=”minikube kubectl --“***

**· 5.Planificación**

En este apartado se usará la herramienta ***Notion*** para mostrar la temporalidad de las fases por las que se ha llevado a cabo el desarrollo y elaboración del proyecto. Toda la información sobre el proyecto se encuentra en el link:

<https://aware-mirror-1cd.notion.site/PROYECTO-ASIR-IES-FERNANDO-WIRTZ-50b662a3fba94430b08b6ef92782dfbb>

**· 5.1. Presupuesto**

[…]

**· 6. Bibliografía**

1. “Es el fin de la infraestructura como servicio tal como la conocemos: esto es lo que sigue” - <https://www.rackspace.com/es/solve/its-end-infrastructure-service-we-know-it>
2. “VMware Workstation Pro 16” - <https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html>
3. “Linux Ubuntu 20.04 Server” - <https://www.stackscale.com/es/blog/ubuntu-20-04-lts/#Ubuntu_Server_2004>
4. “Linux Ubuntu 20.04 Desktop” - <https://www.stackscale.com/es/blog/ubuntu-20-04-lts/#Ubuntu_Desktop_2004>
5. “Linux Mint 20” - <https://blog.redigit.es/que-es-linux-mint-y-como-instalarlo/>
6. “Docker” - <https://www.docker.com/>
7. “Kubernetes” - <https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/>
8. “Netdata” - <https://www.netdata.cloud/>
9. “Helm” - <https://helm.sh/https://www.aplyca.com/es/blog/helm-gestor-de-aplicaciones-para-kubernetes>
10. “Nginx” - <https://kinsta.com/es/base-de-conocimiento/que-es-nginx/>
11. “Let’s Encrypt” - <https://letsencrypt.org/>
12. “Cloudflare” - <https://www.cloudflare.com/es-es/learning/what-is-cloudflare/>
13. “Apache Guacamole” - <https://guacamole.apache.org/>
14. “Lens” - <https://k8slens.dev/>
15. “Freenom” - <https://www.freenom.com/es/aboutfreenom.html>
16. “Tailscale” - <https://tailscale.com/>
17. “Notion” - <https://notionapp.es/descargar-notion/>
18. “GitHub” - <https://github.com/>

[…]