

Bootstrap De Un Clúster De Kubernetes Altamente Disponible En Proxmox Usando Ansible.


Kenneth Miró García

IES Medina Azahara, Córdoba

CFGS Administración de Sistemas Informáticos y Redes

2024

Bootstrap De Un Clúster De Kubernetes Altamente Disponible En Proxmox Usando Ansible.

© 2024 by [Kenneth Miró García](#) is licensed under [CC BY-NC-SA 4.0](#) 

Índice

0.A Abstract.....	3
0.B Resumen.....	4
1 Bootstrap De Un Clúster De Kubernetes Altamente Disponible En Proxmox Usando Ansible.....	5
1.A Objetivos Propuestos.....	7
2 Procedimiento.....	8
2.A Análisis Inicial.....	8
2.A.i Requisitos Deseados.....	8
2.A.ii Evaluación Del Bootstrap Del Clúster.....	8
2.A.iii Recursos Disponibles Finales.....	9
2.B Planificación.....	11
2.B.i Evolución De Las Fases De La Planificación.....	11
Preproyecto.....	11
Investigación Previa.....	11
Puesta En Marcha.....	11
2.B.ii Organigrama Temporal.....	11
Preparación Del Entorno.....	11
Construcción Del Playbook De Ansible.....	11
Despliegue De Pruebas.....	12
Despliegue Definitivo.....	12
Documentación.....	12
2.C Ejecución.....	12
2.C.i Estructura Elegida.....	12
2.C.ii Preparación Del Entorno.....	13
Configuración De Las Máquinas Virtuales.....	15
2.C.iii Construcción Del Playbook De Ansible.....	16
Proceso De Instalación.....	17
El Balanceador De Carga.....	21
2.C.iv Seguimiento Y Control.....	21
3 Resultados.....	23
3.A Objetivos Alcanzados.....	23
3.B Coste.....	23
4 Observaciones.....	25
4.A Consideraciones críticas.....	25
5 Referencias.....	26
6 Documentación Técnica.....	28
7 Glosario.....	35

0.A Abstract

The intention of this project is to bootstrap a fully functional and highly available Kubernetes cluster for educative purposes. Amongst all the evaluated methods to do this, chosen ones were: virtual nodes inside a Proxmox cloud environment in physical nodes within the institution infrastructure for each node in the cluster, to save hardware resources, and Ansible to automatize the procedure on multiple machines, saving time and making it easily replicable. As for Kubernetes itself, various choices were needed to bootstrap said cluster: Calico as CNI for the cluster pod network due to it being widely extended and having a high configuration capability for security purposes, Kube-VIP as the chosen load balancer for the control plane due to it being developed explicitly for Kubernetes, and stacked etcd was the organizational architecture for the control plane members database because it's the approach that saves the most resources.

O.B Resumen

La intención de este proyecto es desplegar un clúster de Kubernetes altamente disponible y completamente funcional que tenga fines educativos en próximos años. Este objetivo se podía plantear y resolver de diversas maneras. Si bien en su planteamiento se buscaba un clúster con máquinas físicas para cada nodo, al final se decidió usar máquinas virtuales en la nube en un entorno de Proxmox montado en varias máquinas físicas dentro del centro, decisión basada en la facilidad de gestión y, principalmente, el ahorro de recursos que supondrá. Para desplegar el clúster se usó Ansible, una tecnología de automatización que permitirá ahorrar tiempo durante el despliegue del clúster y replicarlo en el futuro. Para el propio Kubernetes, que requiere tomar una serie de decisiones en cuanto a su despliegue, se decidieron usar la estructura de etcd apilados, Calico como CNI y Kube-VIP como balanceador de carga.

Keywords: Kubernetes, Calico, Kube-VIP, Proxmox, Ansible, etcd stacked.

1 Bootstrap De Un Clúster De Kubernetes Altamente Disponible En Proxmox Usando Ansible.

Aunque muchas empresas usan tecnologías ya desactualizadas debido a la falta de interés de sus departamentos técnicos o de su departamento ejecutivo, es innegable que en el campo de la administración de sistemas hace falta mantenerse actualizado. Si bien, el sueño de muchos alumnos es encontrar una empresa donde instalen Windows 10 en dos ordenadores a la semana y asistan a usuarios con diversos problemas triviales como encontrar un archivo desaparecido que había acabado en la papelera o ayudar a configurar el idioma de google Drive, otros alumnos pueden aspirar a puestos de más exigencia en empresas más grandes y actualizadas. Y en ambos casos puede que ninguno consiga lo que quiere y el alumnado que deseaba un trabajo de pocas exigencias acabe en una empresa puntera. Es por eso que en esta educación se valora mucho dar una educación de calidad que permita a todo el mundo ser solvente en casi cualquier situación, y saber salir al pase cuando no lo estén.

Ahora bien, una de estas muchas y recientes tecnologías son los contenedores. La tecnología de contenedores, que difiere de la virtualización en que no prealoja los recursos de hardware y hace cada instalación mucho más liviana, está extendiéndose a gran velocidad entre gran multitud de empresas dado que cubren una necesidad fundamental: tener aplicaciones desplegadas rápidamente y muy fácilmente, altamente modularizadas, aisladas del sistema, con un fácil control de versiones. Además, los contenedores abren la puerta al paradigma de la estructura de microservicios, donde cada función de un servicio se atomiza lo máximo para lograr una gran independencia de los nodos, escalabilidad, modularidad y fácil expansión de funcionalidades.

Es cierto que muchas empresas aún usan versiones locales para esto, como Docker,

pero, y en especial para lograr lo segundo, se necesitan orquestadores de contenedores como Kubernetes u OpenShift, la implementación de RedHat de Kubernetes.

Lo que aporta Kubernetes sobre Docker o sus alternativas, tales como Podman, es que Kubernetes funciona en clúster y permite desde unos nodos designados como “control plane” desplegar contenedores en masa, con la posibilidad de aumentar o reducir el número de copias de forma dinámica y en tiempo real, elegir en qué nodos y con qué contenedores puede o no ejecutarse, mantenerlos siempre encendidos realocando contenedores en caso de que su nodo actual se caiga y balancear la carga entre varios contenedores iguales. Un ejemplo muy claro de uso sería una aplicación web altamente modularizada que debe estar siempre activa, o preparar una aplicación en un contenedor que estará ejecutándose en cada ordenador de un alumno para su uso o monitoreo a demanda del profesorado.

Cualquier empresa que desarrolle aplicaciones acabará necesitando personal familiarizado con esta cada vez más común tecnología, más aún las empresas dedicadas a la oferta de servicios de administración de sistemas o de hosting, incluso puede que algunas empresas que quieran tener una aplicación desarrollada a encargo en cada ordenador de algunas secciones necesiten personal familiarizado con Kubernetes si eligen esta tecnología como solución a su problema.

Por todo esto, una educación, si bien no en profundidad necesariamente, pero sí que permita al alumnado familiarizarse con Kubernetes es tan necesaria. Y para facilitar el salto, se ha desplegado este clúster con instrucciones de cómo replicarlo y escalarlo. Además, dada la complejidad de Kubernetes es fácil su relación con varias áreas de estudio de esta FPGS y otras: Desarrollo e Implantación de aplicaciones web, seguridad informática y alta disponibilidad, administración de sistemas, redes, e incluso bases de datos, ya que se

pueden desplegar las bases de datos en forma de contenedores.

Una clara mejora que no ha podido añadirse por falta de recursos y tiempo es la implementación de un sistema de almacenamiento en red compartido entre todos los nodos para que cierta información a la que necesiten acceder todos los contenedores esté siempre disponible. Por supuesto, tanto el alcance del clúster, como su uso y mejoras deben ser considerados a futuro por el profesorado en función del presupuesto disponible y la planificación de los módulos de las FPGS.

1.A Objetivos Propuestos

- Instalar de un clúster de Kubernetes altamente disponible.
- Aprender a elegir un CNI adecuado a las necesidades de la instalación.
- Aprender a elegir según necesidades y circunstancias cada forma de desplegar los elementos del clúster.
- Comprender en profundidad el funcionamiento interno de Kubernetes en su fase de instalación.
- Crear un guión de instalación automático, reutilizable y actualizable.
- Familiarizarse con metodologías de trabajo más cercanas al entorno empresarial.
- Aprender a solventar problemas surgidos durante el despliegue de un clúster de Kubernetes.
- Dejar una infraestructura replicable para el curso próximo con fines educativos.
- Crear una documentación fiable y útil (el presente documento también servirá de documentación).

2 Procedimiento

2.A Análisis Inicial

Durante el planteamiento del proyecto, se tuvieron en mente unas metas, en base a las cuales se plantearon unos requisitos. Si bien las metas cambiaron una vez se comenzó a investigar de forma más exhaustiva tanto la instalación a realizar como los recursos disponibles, estos cambios no han afectado al objetivo final: desplegar un clúster de Kubernetes altamente disponible.

2.A.i Requisitos Deseados

En cuanto a los requisitos iniciales, fueron estos los planteados:

- 2 GB de RAM mínimo en cada nodo.
- Una CPU de al menos 2 núcleos en cada nodo del “control plane”.
- Un SO Linux en cada nodo.
- Un total de 7 nodos:
 - 2 nodos del “control plane”.
 - 4 nodos trabajadores.
 - 1 balanceador de carga.

Sin embargo, la disponibilidad de tantos nodos físicos no solo era complicada, sino menos práctica que usar nodos virtuales en la nube. Estar en la nube en el entorno virtualizado que ofrece un clúster de Proxmox o VMWare ESXi permite hacer snapshots, mudar las máquinas virtuales, gestionar los recursos de hardware más fácilmente, clonarlas, centralizar su gestión, independizarlas de nodos físicos, hacer la infraestructura más escalable...

2.A.ii Evaluación Del Bootstrap Del Clúster

Se comienza investigando lo necesario para bootstrapear un clúster altamente

disponible. Entre las recomendaciones se haya la de tener un número impar de nodos en el control plane para facilitar su proceso de elección de líder, que no se haya descrito en la documentación más accesible de Kubernetes. Además, hace falta elegir una forma de organizar el etcd, la base de datos de Kubernetes con la información de sus miembros, que además debería ser altamente disponible. También hay que elegir qué balanceador de carga usar y qué modo de desplegarlo: la forma tradicional disponiendo de él en un nodo a parte, o una más novedosa desplegándolo junto con el clúster como un contenedor estático usando una IP virtual. El balanceador de carga también debería ser altamente disponible. En cuanto a datos ya conocidos, Kubernetes necesita un CNI para que su red de pods pueda funcionar. Un CNI es el plugin que permite a Kubernetes crear y gestionar una red interna a la que tienen acceso los nodos y en la que se despliegan los contenedores. Son obligatorios, y se decide usar Calico, un CNI con opción comercial y muy extendido.

A su vez, gracias a la experiencia tangencial en el entorno empresarial obtenida durante las prácticas de empresa, se revalora la opción de usar máquinas virtuales en un entorno en nube. Conforme se lleva a cabo esta reevaluación, y en conocimiento de la dificultad de conseguir recursos para este proyecto por parte del centro de estudios, se decide finalmente optar por el enfoque de máquinas virtuales.

2.A.iii Recursos Disponibles Finales

Tras la investigación previa, se deciden usar 3 equipos físicos y montar un clúster de Proxmox VE en ellos. Dichos equipos requieren un mínimo de:

- 8 GB de RAM.
- CPU de 4 núcleos.
- 50GB de espacio libre como mínimo.

Los equipos que se pudieron obtener tienen las siguientes especificaciones:

- Equipo “s0”:
 - 32GB RAM
 - 1TB SSD de disco duro aprox.
 - CPU de 16 núcleos.
- Equipo “s1”:
 - 8GB RAM
 - 100GB SSD de disco duro aprox, totalmente libres en el momento de la cesión.
 - CPU de 4 núcleos
- Equipo “s5”:
 - 16GB RAM
 - 200GB SSD de disco duro aprox, totalmente disponibles en el momento de la cesión.
 - CPU de 12 núcleos.

Los equipos s0 y s1 ya tenían Proxmox VE instalado, al haber formado parte de un clúster preexistente de uso didáctico por parte del tutor del proyecto anteriormente, el cual se deshizo tras la finalización del periodo lectivo. El equipo s0 inició de nuevo el clúster, al que se unió primero s1. El equipo s5 se trata de un ordenador que estuvo disponible una vez finalizado el periodo lectivo, y al que se le instaló Proxmox VE en el mismo momento y se unió al clúster recién creado.

Sin embargo, cabe recalcar que el equipo s1 tenía fallos en su disco duro y tuvo que ser expulsado del clúster, formateado y sufrir, finalmente y cuando se descartaron el resto de causas, un cambio de disco duro SSD. Este nuevo equipo se unió también al clúster una vez finalizado este mantenimiento.

2.B Planificación

La planificación de las fases del proyecto se llevó a cabo en varias fases. Con cada cambio de fase, la planificación variaba una vez conocidas más a fondo las circunstancias bajo las cuales se desarrollaría este proyecto.

2.B.i Evolución De Las Fases De La Planificación

Preproyecto. Aquí se hizo una primera planificación con unos objetivos en mente. Esta fase era más un esquema inicial aproximado.

Investigación Previa. Aquí se hizo una segunda planificación una vez conocidas mejor las características que iba a necesitar el clúster de Kubernetes para funcionar, así como la eliminación de ciertas metas optativas consideradas de interés durante el primer planteamiento en la presentación del preproyecto, pero que una vez más familiarizado con la tecnología y el entorno empresarial se consideraron innecesarias, o como mínimo extremadamente subóptimas.

Puesta En Marcha. Aquí se llevó a cabo la planificación de tiempo definitiva una vez se supo con claridad de qué equipos se dispone y se tenía la investigación previa finalizada.

2.B.ii Organigrama Temporal

El proyecto final se dividió en etapas de ejecución. Estas fueron:

Preparación Del Entorno. En esta primera etapa se crea el clúster de Proxmox VE que servirá de base y se crean las máquinas virtuales que serán sus nodos. Se le asignaron 4 horas de duración.

Construcción Del Playbook De Ansible. En esta segunda fase se usará de plantilla un playbook previo que elaboré con mis primeros pasos en Kubernetes durante el curso escolar por iniciativa propia. El uso de este recurso preexistente adaptándolo a las nuevas

circunstancias facilitó en gran medida el trabajo ahorrando mucho tiempo. Se le asignaron entre 12 y 15 horas de duración.

Despliegue De Pruebas. Tercera etapa donde se lanzaron los playbooks en una máquina aislada con el fin de comprobar que todo funcionaba correctamente, en especial ciertas instrucciones que tuvieron que ser improvisadas al no aparecer en ninguna fuente de documentación. Dichas instrucciones, que serán explicadas más en profundidad en futuras secciones, son construir un manifiesto de pod estático usando CRI-O y podman. Se le asignaron de 2 a 5 horas de duración.

Despliegue Definitivo. Última etapa, donde se lanza el playbook en todas las máquinas finales. En este punto se realizan snapshots de cada máquina previo al despliegue, y se comprueba que, efectivamente, la instalación se efectúa de forma exitosa. En caso negativo, se carga la snapshot y se corrige el playbook antes de ejecutarlo de nuevo. Se le asignaron de 2 a 5 horas de duración.

Documentación. Se decidió que la documentación se elaboraría de forma paralela.

2.C Ejecución

Una vez todo listo, se comienza con la ejecución de cada etapa. Lo primero es explicar la estructura elegida.

2.C.1 Estructura Elegida

Kubernetes divide sus miembros en dos grupos: control plane, donde se encuentran los nodos con los componentes de Kubernetes necesarios para su gestión, y los nodos trabajadores que contienen lo necesario para poder ejecutar los pods, siendo un pod una forma de organizar contenedores. Generalmente un pod solo contiene un contenedor, pero puede contener varios en ciertos casos.

En cada uno de los 3 nodos físicos del clúster de Proxmox VE se instalarán 2 máquinas virtuales: una estará dentro del control plane de Kubernetes y la otra será un nodo trabajador. De esta forma, si un nodo físico cae, solo se pierden un trabajador y un miembro del Control plane, asegurando que siempre estén operativos al menos un nodo de cada tipo y garantizando la operatividad del clúster.

En cuanto al balanceador de carga y el etcd, se usarán el enfoque de contenedores estáticos con IP Virtual para el primero y el enfoque de etcd apilado para el segundo. En ambos casos esta elección reduce el coste en recursos, facilita la alta disponibilidad de dichos elementos y ahorra tiempo de instalación. Debido a esto, solo estos 6 nodos virtuales mencionados anteriormente son necesarios.

Para el sistema operativo se ha elegido CentOS Stream 9, por su robustez como SO de servidores y su gran similitud con RedHat, una distribución comercial y empresarial de Linux muy extendida. Es un SO libre que intenta ser lo más parecido posible a la distribución ya mencionada. Además, gran parte de las tecnologías usadas aquí son o bien desarrolladas o bien respaldadas por RedHat, lo que hace su instalación mucho más sencilla y compatible en CentOS, que si bien no fue un factor decisivo a la hora de elegir este SO sobre otros, sí que es un buen añadido.

2.C.ii Preparación Del Entorno

Primero y una vez elegidos los equipos se crea el clúster de Proxmox en los equipos s0 y s1. Luego, se instala Proxmox VE en el equipo s5. Durante la instalación surge la primera inconveniencia: Proxmox VE no tiene drivers de gráficas nvidia durante el proceso de instalación, por lo que aunque eventualmente los instale, el instalador no puede ser ejecutado sin aplicar alguna solución. Tras una investigación, se decide probar con una

solución común: entrar en el modo avanzado de instalación de Proxmox y editar la entrada que inicia el asistente para añadir la opción *“nomodeset”* en la línea que empieza por *“Linux”*.

Una vez la instalación estuvo completa, se pasó la iso de CentOS Stream 9 al nodo s0, donde se crearon las dos primeras máquinas virtuales, que harán de nodo del control plane y nodo trabajador, con los nombres k-master-1 y k-worker-1.

Se intentaron clonar a otros nodos, pero debido a la configuración de Proxmox si los otros nodos no tienen almacenamiento en red compartido no es posible hacer el traslado, por lo que se procede a copiar la iso en sendos nodos s1 y s5 para proceder con la instalación de las máquinas virtuales, que sigue sin problemas en el nodo s5 bajo los nombres k-master-3 y k-worker-3. En el nodo s1, por otro lado, la instalación no pudo llevarse a cabo ya que no tenía acceso a su disco duro para las máquinas virtuales. Las diversas soluciones intentadas, por orden menor a mayor coste en tiempo, llevaron a formatear el nodo s1, pero el error persistía y además ahora aportaba información adicional del mismo, gracias a lo cual se valoró que pudiera estar dañado. De nuevo, aplicando soluciones de menor a mayor coste en tiempo se probó a cambiar el disco duro por otro distinto, confirmando el diagnóstico.

Una vez estuvo de nuevo operativo el s1, se unió al clúster otra vez, ante lo cual surgió otro problema: el nodo s1 no salió adecuadamente del clúster y no podía expulsarse desde la interfaz web. Tras una exhaustiva búsqueda se encontró una solución que no implicaba destruir la base de datos de miembros del clúster. La línea de comandos de Proxmox permite muchas más acciones que la interfaz web, siendo una de estas indicarle cuántos nodos se esperan que estén activos. Al no tener comunicación con el antiguo nodo

s1, no puede expulsarlo del clúster, y al estar en la base de datos de miembros el nuevo nodo s1 no puede unirse tampoco. Sin embargo, debido a que el nodo s1 está inactivo dentro del clúster, se puede indicarle al nodo s0 que se esperan 2 nodos en el clúster, por lo que forzosamente borra la información del nodo inactivo, permitiendo al recién formateado s1 unirse de nuevo.

Tras todo este proceso, se crean y configuran también las máquinas virtuales con CentOS Stream 9 en el nodo s1, de nombres k-master-2 y k-worker-2.

Configuración De Las Máquinas Virtuales. Todas las máquinas trabajadoras tienen 2GB de RAM y 1 núcleo de CPU. Todas las máquinas del control plane tienen 3GB de RAM y 2 núcleos de CPU. Todas las máquinas del s0 tienen 35GB de espacio en disco, todas las máquinas del s1 tienen 20GB de espacio en disco y todas las máquinas del s5 tienen 25GB de espacio en disco. Los motivos de esto son que el s1 tenía menos espacio en disco disponible y por un error humano se asignó de forma automática el espacio en disco de las máquinas en el s0, siendo el espacio de las máquinas del s5 el deseado. Todas las cuentas root están habilitadas con acceso por ssh deshabilitado, y en todas las máquinas se ha creado un usuario kubernetesadmin con permisos de administrador. En ninguna máquina hay partición swap debido a que Kubernetes no sabe gestionarla, y aunque actualmente el soporte swap está en beta se ha preferido actuar sobre seguro. Todas las máquinas tienen como puerta de enlace la IP 192.168.8.1 y como servidores DNS las IPs 192.168.8.1, 192.168.1.9 y 8.8.8.8. Todas las máquinas están instaladas en modo servidor sin GUI con los complementos “Console Internet Tools”, “Performance Toos” y “Debugging Tools” por la utilidad que ofrecen. En cuanto a las IPs, las máquinas virtuales tienen las siguientes:

- k-master-1: 192.168.9.121/23

- k-worker-1: 192.168.9.141/23
- k-master-2: 192.168.9.122/23
- k-worker-2: 192.168.9.142/23
- k-master-3: 192.168.9.123/23
- k-worker-3: 192.168.9.143/23

Al final se usó más tiempo del estimado en esta fase debido a los múltiples problemas encontrados.

2.C.III Construcción Del Playbook De Ansible

Ansible utiliza un tipo de archivo denominado playbook escrito en lenguaje yaml para guionizar sus acciones. Se tomó de plantilla un playbook más simple que usé durante el curso para ahorrar tiempo. En esta ocasión, se dividió el contenido en distintos playbooks, uno para cada fase del bootstrap del clúster, siendo estas pre-requisitos, generación de manifiestos, bootstrap fase 1, bootstrap fase 2 y unión de los trabajadores. Todos están dentro de un mismo directorio, dentro del cuál hay un fichero de texto plano con instrucciones y otros dos subdirectorios. Estos subdirectorios contienen los ficheros con las variables utilizadas durante el bootstrap del clúster por ansible y los archivos de plantilla en formato jinja, que ansible copia a las máquinas objetivo sustituyendo las variables indicadas dentro del archivo por el contenido que ansible tiene en memoria. A su vez, hay un inventario de plantilla. Se recomienda usar este inventario de forma explícita al ejecutar los playbooks a menos que el nodo que está ejecutando los playbooks sea definitivo.

Este playbook se ejecutó desde mi ordenador personal para ahorrar recursos al centro, pero este directorio se copió junto con las instrucciones y las explicaciones a un ordenador del centro y otra copia le fue entregada al tutor del proyecto. Para permitir la

conexión de ansible, se generó una clave ssh que se copió a todos y cada uno de los nodos.

Proceso De Instalación. Estos pasos se realizan en todas las máquinas a menos que se indique lo contrario. Lo primero de todo es activar el modo de SELinux a permisivo. Set Enforce Linux es un módulo de Linux que tienen muchas distribuciones, entre las que se encuentran CentOS y RedHat, que aporta mucha seguridad al núcleo de Linux, pero también es muy restrictivo. Sus tres modos son Permissive, Enforced y Disabled. En modo Enforced bloquea y registra todas las acciones no permitidas, y en modo Permissive solo las registra, por lo que en lugar de deshabilitarlo siempre es mejor opción ponerlo en modo permisivo. Una alternativa es añadir a la lista de acciones permitidas todas las posibles acciones de Kubernetes, pero eso solo ya escapa al alcance de este proyecto.

Tras ello, se añaden los repositorios de Kubernetes y CRI-O, el Container Engine elegido para que Kubernetes ejecute los contenedores. Este Container Engine fue elegido ya que está creado específicamente para cumplir con los requisitos de Kubernetes y facilita muchísimo la instalación y ahorrando problemas de compatibilidad. Sin embargo, también tiene sus problemas, ya que a diferencia de otros Container Engines de por sí solo no hace mucho, siendo recomendable usar Podman, Containerd o Docker en su lugar de ser necesario tener una versión standalone de un Container Engine para ejecutar contenedores de manera local. Por este motivo hubo que realizar pruebas posteriores antes de levantar el clúster definitivo.

Luego, desactivo la memoria swap. En este caso no es necesario, pero aún así lo incluyo para facilitar su implantación. El siguiente paso es actualizar las máquinas e instalar kubeadm, la utilidad para bootstrapear el clúster. Adicionalmente, deben modificarse ciertos archivos y variables del sistema para permitir el enrutamiento de Kubernetes.

Una vez hecho eso, en los nodos donde se deseen ejecutar contenedores se instalan las dependencias de CRI-O, y posterior a ello el propio CRI-O. Por último, se instala kubelet, el agente que gestiona los contenedores en los nodos trabajadores según las órdenes del control plane, y se habilitan tanto CRI-O como kubelet. En el caso del presente proyecto, se desean poder ejecutar contenedores en todos los nodos, por lo que estos pasos se realizan simultáneamente en todas partes. Adicionalmente se desactiva y deshabilita el cortafuegos nativo de CentOS, Firewalld. Si bien este se puede configurar para permitir todas las conexiones necesarias de Kubernetes, debido a su incomodidad de uso para ciertas tareas más complejas, y afirmo estando muy familiarizado con él, he decidido apagarlo y recomendar en su lugar configurar posteriormente IPTABLES, aunque también se puede reactivar en caso de así desearlo.

Finalmente y para terminar con los pre-requisitos, en los nodos del control plane se instala kubectl, la utilidad de consola para controlar Kubernetes. Este componente también puede instalarse en otra máquina que no forme parte del control plane con el fin de poder usarla a distancia, ya que en realidad el funcionamiento interno de Kubernetes consiste en un servicio denominado kube-api que es el encargado de gestionar todos los componentes, y kubectl se conecta a kube-api y le envía sus órdenes, por lo que si bien no es obligatoria su instalación en los nodos del control plane, es altamente recomendable.

Luego, se reliza la siguiente fase, donde en un solo nodo del control plane se genera un manifiesto: un tipo de archivo que Kubernetes lee en cada inicio y lo usa para levantar un contenedor. Este manifiesto es el que contiene la información de Kube-VIP, el balanceador de carga desarrollado por Kubernetes y que además expande sus funcionalidades. Fue elegido este software debido a la compatibilidad y a la mejora de funcionalidades que

ofrece, aunque otras soluciones se hayan mejor documentadas y más fáciles de ejecutar.

Este manifiesto se puede generar de dos formas: manualmente o copiando una plantilla, o generando uno automáticamente con una orden al Container Engine elegido, CRI-O en este caso. Este último método fue el elegido para asegurar compatibilidad, pero a su vez generó problemas debido a la falta de documentación sobre como ejecutar este mismo procedimiento con CRI-O.

Ahora sí, se procede a la última fase de bootstrap del clúster. Primero, en ese mismo nodo maestro, ese inicializa el clúster pasando la dirección de red interna para los pods de Kubernetes y la IP del balanceador de carga, que es necesario para que la alta disponibilidad del clúster funcione debidamente. Para la inicialización se usa el argumento `--upload-certs` que sube los certificados del control plane al resto de nodos del control plane, paso que se debe hacer manualmente si no usas esa opción. Los certificados que se generan de forma automática durante este paso duran 2 horas. Si tras ese periodo se desea añadir nuevos miembros al clúster, habrá que generar nuevos certificados. También se recomienda refrescar los certificados periódicamente y mantenerlos dentro de los Secrets, ya que contienen información delicada. Tras ello, se genera un archivo denominado KUBECONFIG, que contiene configuración del clúster. Se usa el archivo de configuración de ejemplo de kubernetes, que debe copiarse a la ubicación debida.

Tras ello, se agrega un CNI: un tipo de plugin que Kubernetes requiere para gestionar su red interna de pods. Como ya se mencionó, Calico fue el CNI elegido para esta tarea debido a su enfoque empresarial, a ser ampliamente utilizado en entornos profesionales y a su gran capacidad de configuración a nivel de seguridad, aunque gran parte de estas funciones están tras una barrera de pago. Hay otras alternativas también con

la misma calidad de prestaciones, pero para este supuesto, este es suficiente. Para su instalación hace falta un pre-requisito: el triggera operator, y luego un archivo yaml con la información relevante para que Kubernetes instale el plugin. Por último, se instalan los binarios de calicoctl, la línea de consola para tener un mejor control de Calico y su configuración.

Ahora, se copia el mandato de unión al resto de nodos que formarán parte del control plane y se ejecuta. Para ello se registra este comando en una variable de ansible, que luego se globaliza para que aplique a otras máquinas objetivo. Además, también se instalan los binarios de calicoctl en estas máquinas, se copia el manifiesto del balanceador de carga para que sea altamente disponible. Opcionalmente, también se puede ejecutar un playbook para eliminar la etiqueta de los nodos maestros que les impide ejecutar pods dentro, de forma que en caso de ser necesario se pueden ejecutar pods de la red de Kubernetes también en el control Plane. Esta decisión requiere de mayor configuración a la hora de levantar cada pod si este no es deseable que se ejecute en los nodos del control plane. Cabe destacar que esto no es necesario para los contenedores estáticos de Kube-VIP ya que estos están fuera de la red de Kubernetes, con las ventajas y desventajas que ello conlleva. Sigue siendo deseable que este balanceador de carga esté activo dentro de la red de Kubernetes, tarea para la que hay solución: lanzar un servicio en Kubernetes usando como imagen este balanceador de carga, y desactivando los contenedores del control plane para evitar conflictos de IP. Sin embargo, esto presenta ciertos problemas: en caso de caída total del clúster este no podría volver a levantarse ya que no quedarían balanceadores de carga para poner Kubernetes a funcionar.

Para el etcd en modo apilado o stacked no hacen falta pasos adicionales, es el modo

por defecto.

Y en último lugar, copio el mandato de unión de nodos trabajadores por el mismo proceso anterior y lo ejecuto en los nodos trabajadores.

El Balanceador De Carga. Kube-VIP se ha ejecutado por IP virtual en forma de contenedor estático en cada nodo del control plane usando el protocolo ARP. Esto significa que en cada nodo hay un contenedor con una IP idéntica a todos, pero solo una de estas IPs es accesible desde el exterior: el líder. Si el líder cae, un nuevo líder es elegido. Ya que todos comparten la misma IP, acaba resultando siendo altamente disponible. El otro modo de ejecutarse es por protocolo de nivel 3, BGP.

2.C.iv Seguimiento Y Control

Para solucionar principalmente cualquier problema con el bootstrap, se realizó un despliegue de pruebas en una máquina virtual aislada antes de replicar cada paso en el clúster para comprobar si, efectivamente, este se desplegaba adecuadamente.

El primero de los problemas surgió con el formato del inventario, de solución fácil, y el siguiente consistió en pasar el usuario de conexión, error que nunca antes me había ocurrido, pero también de solución fácil. Los siguió un problema al usar los mandatos sudo, problema que tampoco tuve nunca previamente por haberme conectado siempre a la cuenta root. Tras arreglarlo, los prerequisites se pudieron aplicar sin problemas.

Con la generación del manifiesto empezaron a surgir problemas y todos ellos relacionados con ansible. Usando los módulos built-in de Ansible, el método recomendado por Ansible, no se podía ejecutar debidamente el mandato para generar el manifiesto, por lo que al final la solución elegida tras mucha prueba y error fue ejecutar un script. Si bien Ansible permite ejecutar un script de la máquina local en la máquina objetivo, se prefirió

usar una plantilla de script para que así las variables de ansible pudieran seguir aplicándose y con ello manteniendo la centralización de la gestión de la instalación.

Durante el faso de inicialización del clúster ciertos errores empezaron a ocurrir con el servicio kubelet, que a pesar de tener un archivo de configuración, no era capaz de leerlo, y por ende fallando en su inicio. Aquí es donde tener snapshots resulta una gran ventaja, permitiendo recargarla y solucionando así los problemas presentados. Sin embargo, ese no resultó ser el problema. Este parecía residir en alguna configuración errónea de las generadas automáticamente que impedía a Kubernetes inicializar correctamente. Al final, se descubrió la fuente del error: el componente de Kubernetes que controla el servidor no era accesible a través del Proxy a menos que se configurase mejor. Fue muy difícil de solventar, y debe tenerse en cuenta que según diversas fuentes debido a la naturaleza aún bastante experimental de Kubernetes es frecuente que en ocasiones las instalaciones fallen y tenga que hacerse troubleshooting.

Tras ello, el resto del clúster pudo inicializarse sin problemas.

3 Resultados

El clúster fue desplegado correctamente, y los objetivos logrados. El resultado final de los playbooks de Ansible es diferente a lo que se planteó en un primer momento, pero era esperable.

3.A Objetivos Alcanzados

- Instalar de un clúster de Kubernetes altamente disponible.
- Aprender a elegir un CNI adecuado a las necesidades de la instalación.
- Aprender a elegir según necesidades y circunstancias cada forma de desplegar los elementos del clúster.
- Comprender en profundidad el funcionamiento interno de Kubernetes en su fase de instalación.
- Crear un guión de instalación automático, reutilizable y actualizable.
- Familiarizarse con metodologías de trabajo más cercanas al entorno empresarial.
- Aprender a solventar problemas surgidos durante el despliegue de un clúster de Kubernetes.
- Dejar una infraestructura replicable para el curso próximo con fines educativos.
- Crear una documentación fiable y útil (el presente documento también servirá de documentación).

3.B Coste

En la práctica, no hubo coste ya que todo se realizó con equipos de sobra en el instituto y software gratuito, y en gran parte de código abierto. En la realidad, de querer

replicarse esto, tras una búsqueda rápida y poco exhaustiva, el coste de 3 equipos de al menos 4 núcleos de CPU, 8GB de RAM y 100GB de disco duro rondaría 285€ aproximadamente.

4 Observaciones

Desplegar un clúster de Kubernetes lleva poco tiempo si no da errores, pero la configuración necesario para hacerlo no es por ello menos compleja. Al final, el método más eficaz es crear scripts en cualquier tecnología de automatización y ejecutarlos. Esta tecnología es propensa a errores durante su instalación, y si bien se ha usado la herramienta kubeadm para hacer el despliegue, toda esta configuración se puede hacer de forma manual.

4.A Consideraciones críticas

Dado que es común que Kubernetes de errores durante el despliegue, es importante saber dónde buscarlos. Guardar el output del comando de inicialización ha resultado tremendamente útil, así como el mandato “kubeadm reset” para deshacer el clúster. También es crucial entender el proceso que realiza kubeadm para hacer bootstrap del clúster, así como saber buscar en los logs cuál es la causa.

Es extremadamente recomendable establecer un servidor DNS para el clúster de Kubernetes, y especificar las IP durante la fase de Bootstrap del clúster por nombre DNS en lugar de por IP.

5 Referencias

Documentación Oficial De Kubernetes. (Abril, Mayo y Junio 2024).

- Crear un clúster de Kubernetes con Kubeadm:
 - <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/>
- Alta disponibilidad con Kubeadm:
 - <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/>
- Opciones de software para balance de carga:
 - <https://github.com/kubernetes/kubeadm/blob/main/docs/ha-considerations.md#options-for-software-load-balancing>
- Kube-VIP como pod estático:
 - <https://kube-vip.io/docs/installation/static/>
- Inicio rápido de Calico:
 - <https://docs.tigera.io/calico/latest/getting-started/kubernetes/quickstart>
- Instalación de calicoctl:
 - <https://docs.tigera.io/calico/latest/operations/calicoctl/install>
- Documentación de Ansible:
 - Módulos:
 - https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html
 - Inventario:
 - https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html

- Variables:
 - https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html

Foros de github, stackoverflow y otros para las búsquedas de soluciones a problemas.

6 Documentación Técnica

Configuración y creación del clúster:

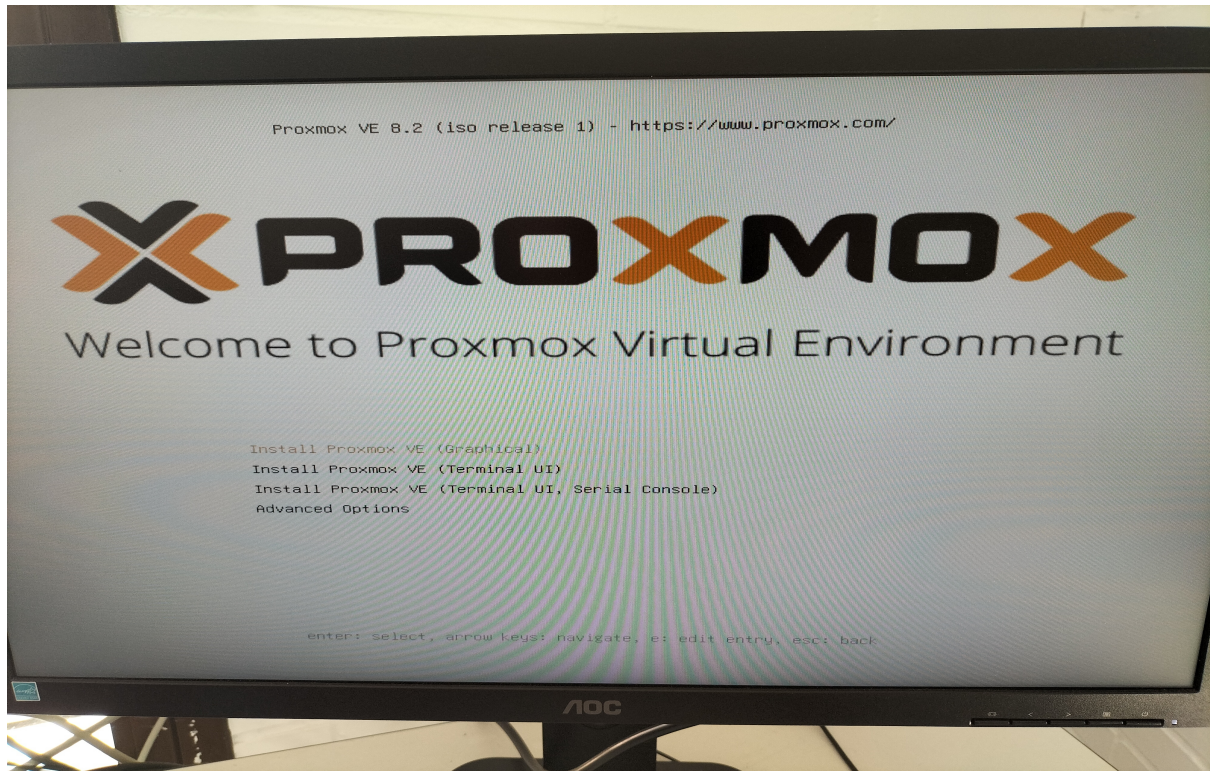


Figure 1: Instalación de Proxmox VE

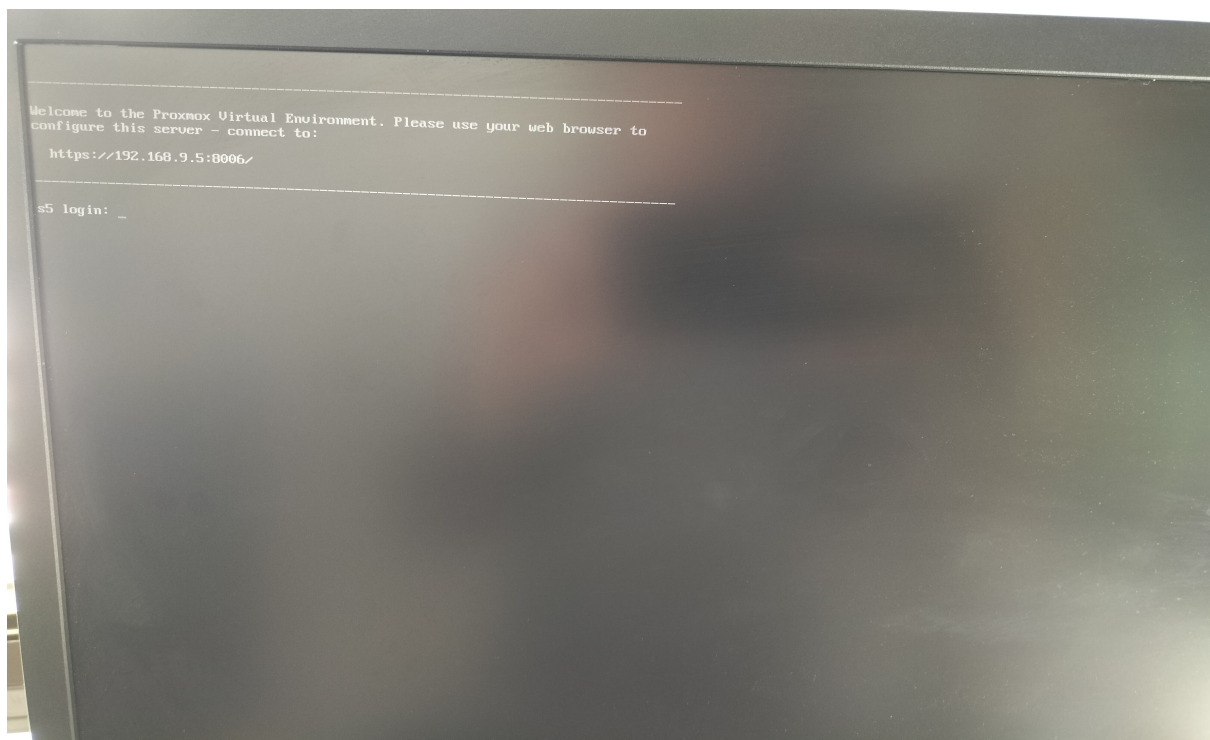


Figure 2: s5 Instalado

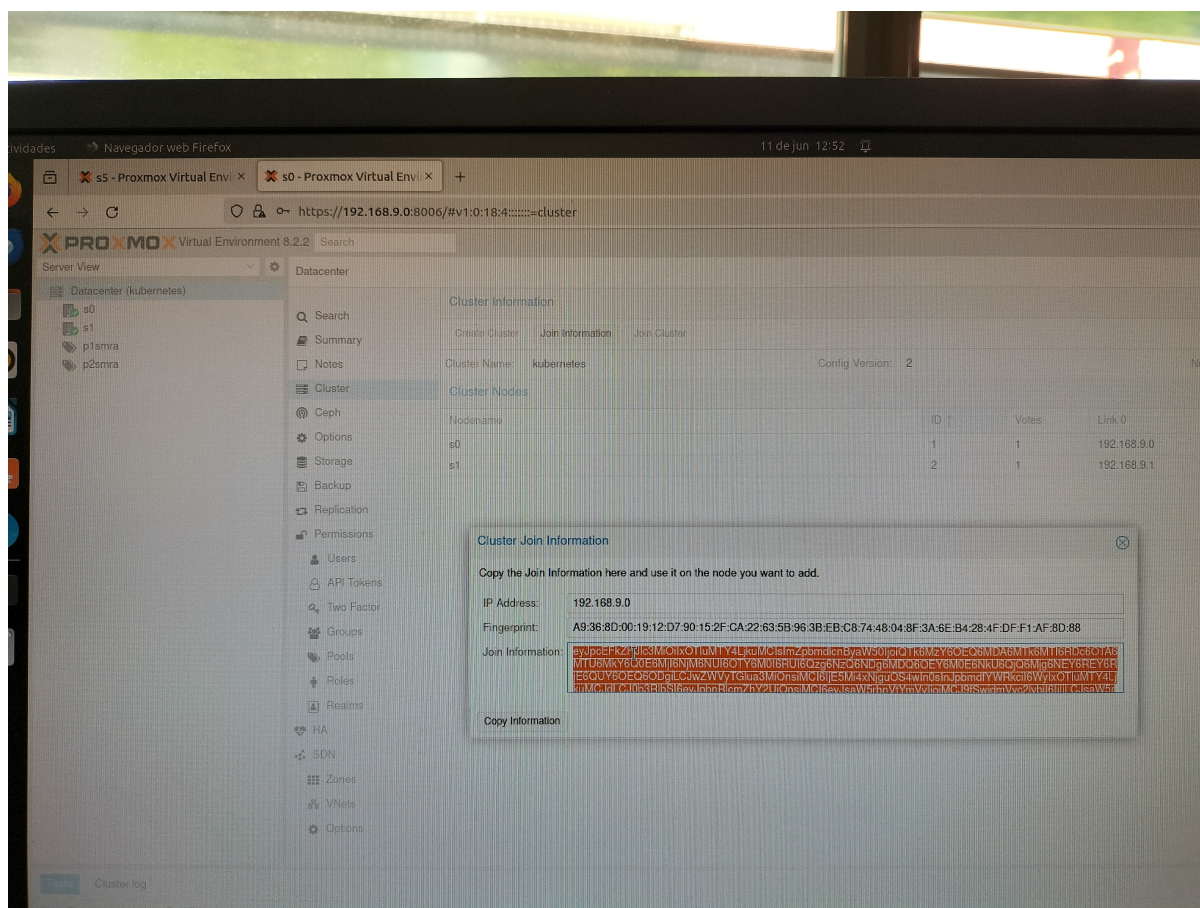


Figure 3: Clúster de Proxmox de máquinas pre-existentes

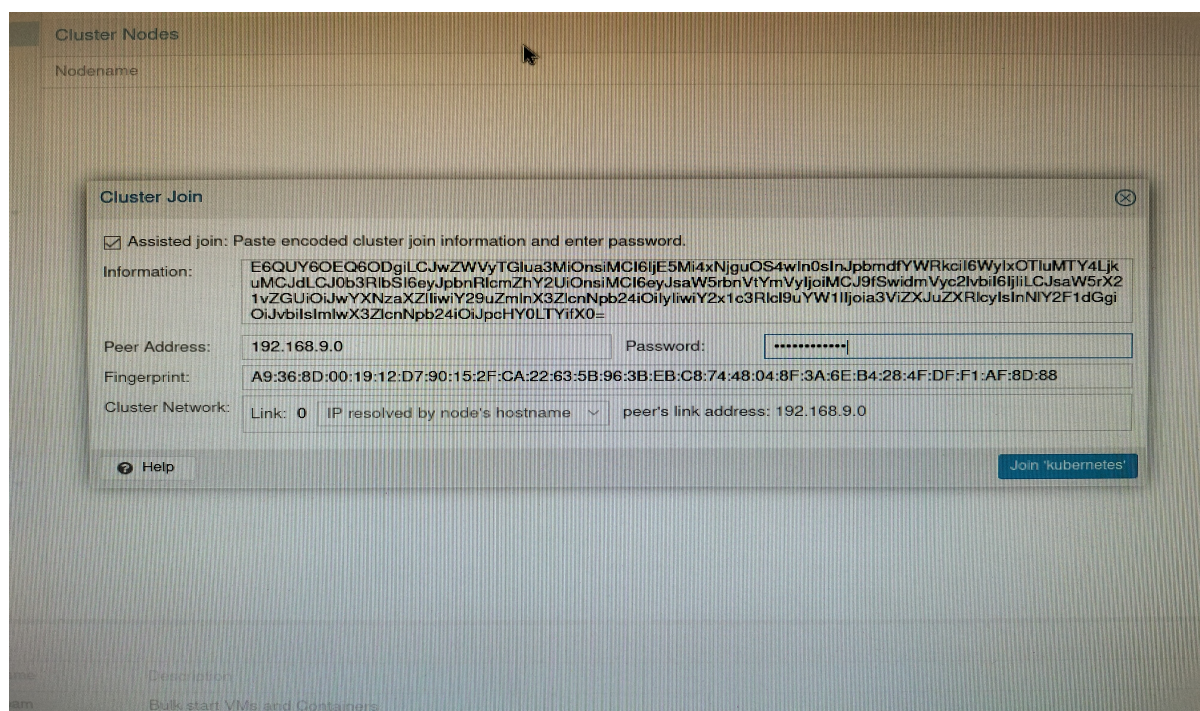


Figure 4: s5 Uniéndose Al Clúster

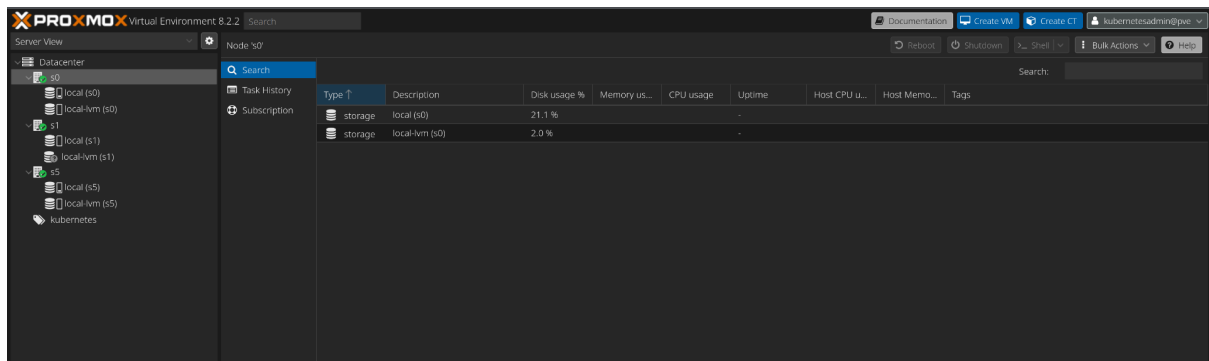


Figure 5: Clúster Completo y Usuario Añadido

```
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Jun 14 11:35:49 CEST 2024 on pts/0
root@s0:~# pvecmd delnode s1
Killing node 2
Could not kill node (error = CS_ERR_NOT_EXIST)
root@s0:~# pvecmd nodes

Membership information
-----
Nodeid      Votes Name
    1         1 s0 (local)
    3         1 s5
root@s0:~# pvecmd expected 2
root@s0:~# pvecmd delnode s1
Node/IP: s1 is not a known host of the cluster.
root@s0:~#
```

Figure 6: Problema De Membresía Del s1 Solucionado

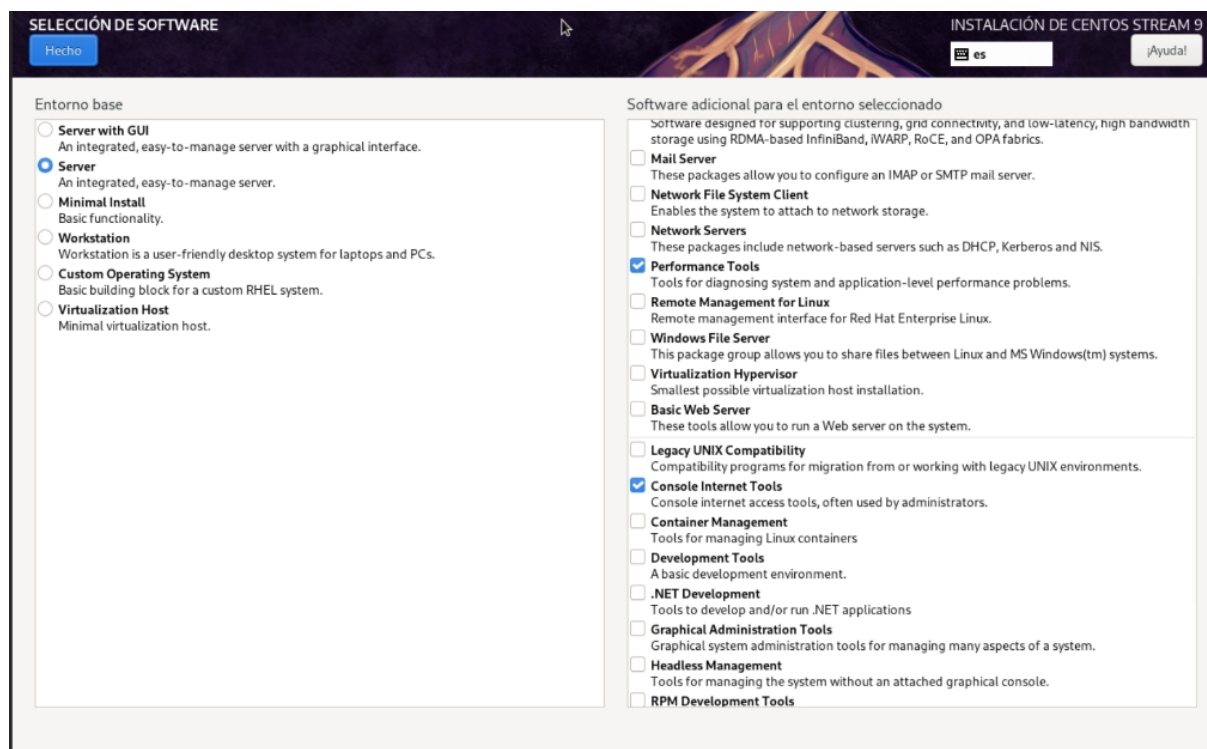


Figure 7: Características De Instalación De Las Máquinas Virtuales

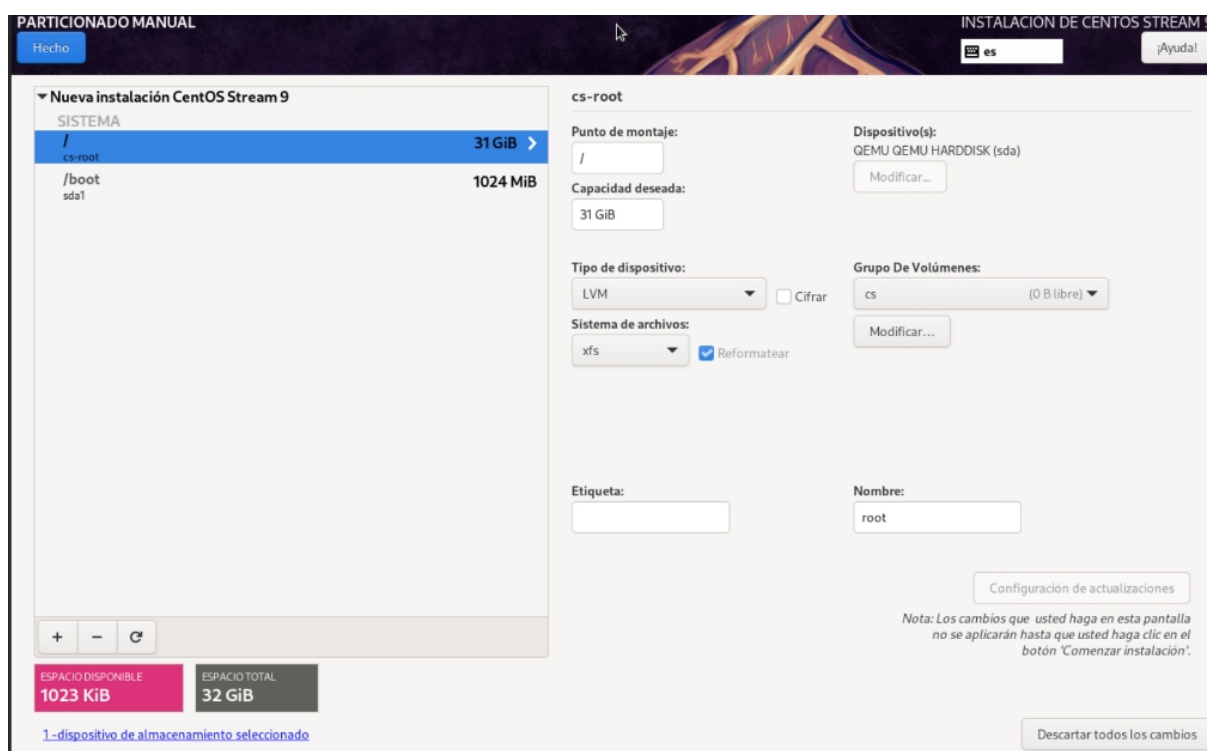


Figure 8: Discos Duros De Las Máquinas Virtuales

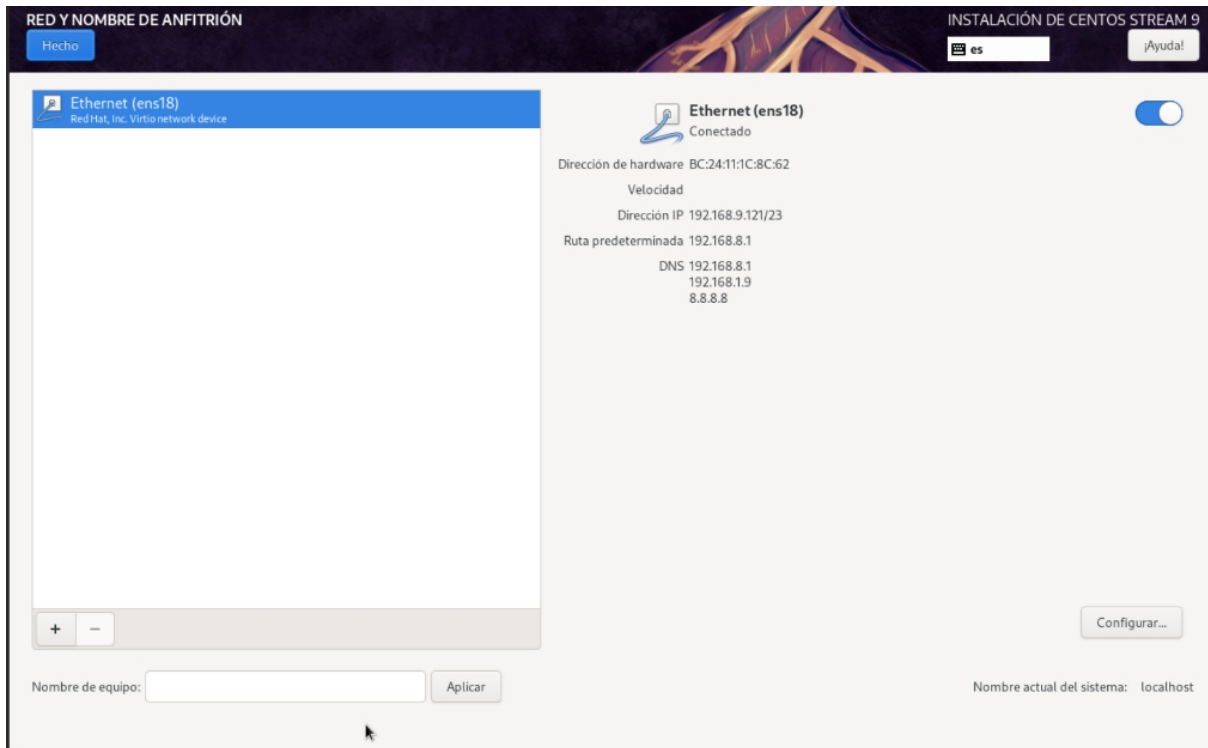


Figure 9: Configuración De Red De Las Máquinas Virtuales

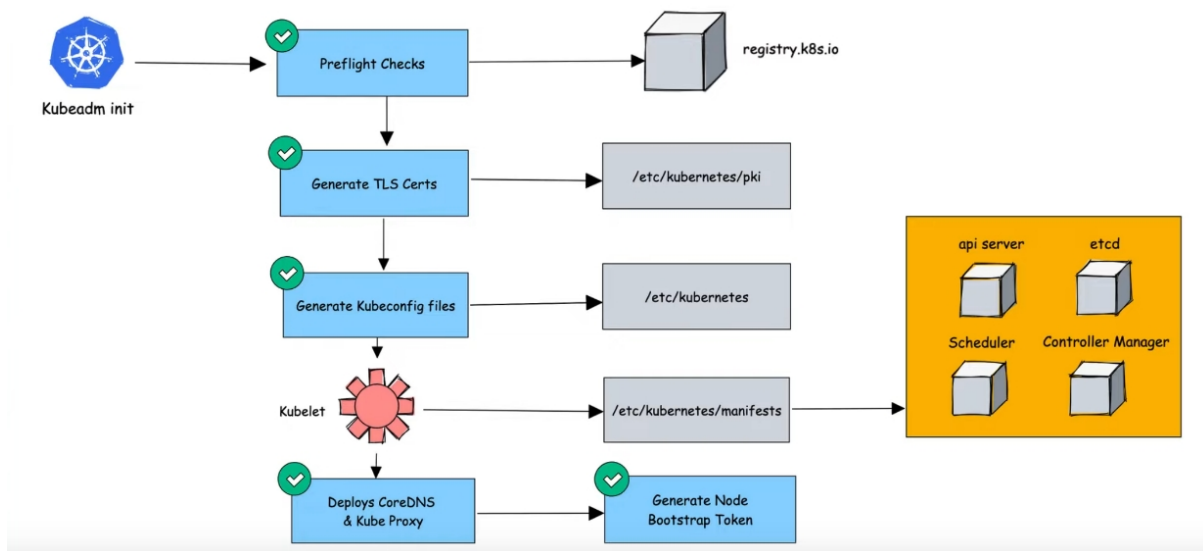


Figure 10: Proceso De Bootstrap Realizado Por kubeadm

- pre-requisites.yaml
 - <https://github.com/badchoiceskenny/Bootstrap-Kubernetes-on-CentOS-Ansible-Playbooks/blob/main/pre-requisites.yaml>
- control-plane-manifests.yaml
 - <https://github.com/badchoiceskenny/Bootstrap-Kubernetes-on-CentOS-Ansible-Playbooks/blob/main/control-plane-manifests.yaml>
- control-plane-init-1.yaml
 - <https://github.com/badchoiceskenny/Bootstrap-Kubernetes-on-CentOS-Ansible-Playbooks/blob/main/control-plane-init-2.yaml>
- control-plane-init-2.yaml
 - <https://github.com/badchoiceskenny/Bootstrap-Kubernetes-on-CentOS-Ansible-Playbooks/blob/main/control-plane-init-2.yaml>
- worker-noder-join.yaml
 - <https://github.com/badchoiceskenny/Bootstrap-Kubernetes-on-CentOS-Ansible-Playbooks/blob/main/worker-nodes-join.yaml>
- instrucciones.txt
 - <https://github.com/badchoiceskenny/Bootstrap-Kubernetes-on-CentOS-Ansible-Playbooks/blob/main/instrucciones.txt>
- Directorio files con los archivos plantilla
 - <https://github.com/badchoiceskenny/Bootstrap-Kubernetes-on-CentOS-Ansible-Playbooks/tree/main/files>
- Directorio vars con los archivos de variables
 - <https://github.com/badchoiceskenny/Bootstrap-Kubernetes-on-CentOS-Ansible-Playbooks/blob/main/vars>

[Ansible-Playbooks/tree/main/vars](#)

7 Glosario

Ansible: software de automatización.

Calico: CNI de Kubernetes.

Cillium: CNI de Kubernetes.

CNI: plugin obligatorio de Kubernetes que le permite gestionar la red interna de los pods, que no será accesible desde el exterior.

Container Engine: software que es capaz de ejecutar contenedores.

Contenedor estático: contenedor que levanta Kubernetes pero fuera de su red, por lo que no es capaz de vigilarlo.

Control plane: nombre que da Kubernetes al conjunto de todos los nodos que tengan los componentes de control de Kubernetes.

CRI-O: un Container Engine.

Etcad: componente de Kubernetes con la base de datos de miembros.

Kubeadm: componente opcional de Kubernetes que levanta el clúster y permite unir miembros.

Kube-apiserver: componente de Kubernetes que expone la API, es decir, su front-end.

Kube-controller-manager y cloud-controller-manager: Cada uno es un conjunto de componentes que se ejecutan como un único proceso. El primero controla y gestiona contenedores y el segundo controla enrutamiento y balanceo de carga.

Kubectcl: línea de comandos que se comunica con el apiserver para controlar kubernetes.

Kubelet: agente de Kubernetes que ejecuta los contenedores.

Kubeproxy: Permite a los contenedores designados ser accesibles desde el exterior.

Kube-scheduler: componente de Kubernetes que comprueba que los pods están en el estado deseado.

Kube-vip: software diseñado para hacer de balanceador de carga para los nodos del control plane y aumentar sus funcionalidades para los servicios en caso de así desearse.

Servicio (contexto de Kubernetes): una forma de desplegar a los pods, que incluye entre otros balanceo de carga si así se especifica y acceso desde la red externa.

Stacked etcd: forma de organizar el etcd en donde este se encuentra dentro de cada nodo del control plane. Su otra forma es externo, guardándolo en otros nodos dedicados para ello.

Virtual IP: IP asignada dentro de un entorno virtual de un nodo. Estas IP no son accesibles desde el exterior, solo desde el nodo local, y para ser accesibles de forma externa deben o exponer su ip por un puerto o hacer redirección de puertos.