Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Desarrollo y Operaciones (DevOps)

Automatización del despliegue de infraestructura en un clúster de Kubernetes mediante el uso de GitOps

Trabajo fin de estudio presentado por:	Jonatan Ruedas Mora
Tipo de trabajo:	Trabajo Fin de Máster
Director/a:	Rafael Merlo Loranca
Fecha:	2021 - 2022

Resumen

Este documento recoge los detalles del proceso llevado a cabo para la consecución de los objetivos del Trabajo de Fin de Máster, que comprende el estudio, diseño e implementación de un flujo de GitOps a fin de automatizar y simplificar el despliegue de aplicaciones y de infraestructura sobre un clúster de Kubernetes.

Al igual que un cambio en el código de una aplicación dispara un proceso para que esta se empaquete y se despliegue, GitOps permite que el despliegue y aprovisionamiento de las aplicaciones e infraestructura se aplique de manera automática al modificar los ficheros de estado y/o configuración que se encuentran bajo control de versiones. Como consecuencia, se mejora la auditabilidad del entorno, se facilita la restauración del estado anterior y se consigue trazabilidad de los problemas.

El objetivo de este Trabajo de Fin de Máster es el de estudiar el funcionamiento de GitOps para validar su grado de madurez a fin de llevar a cabo su implementación en un entorno corporativo aplicando así las mejores prácticas de trabajo sobre Kubernetes. Además, se pretende evaluar si permite reducir el que tiempo que los equipos de operaciones dedican a investigar y solucionar problemas en los entornos de manera manual.

Palabras clave: GitOps, Kubernetes, DevOps, CI/CD, Cloud

Abstract

This document brings together the details of the process conducted to meet the goals of the master's degree final project. These objectives include the study, design, and implementation of a GitOps workflow that automates the deployment of applications and infrastructure on a Kubernetes cluster.

Just as a change in an application's code triggers a process that will package and deploy that application. GitOps allows the deployment and provisioning of applications and infrastructure automatically by modifying the state or configuration files that are under version control. Therefore, the auditability of the environment is enhanced, the rollback process becomes simple and the traceability of the root cause of the problems is achieved.

The goal of this master's degree final project is to study the operation of GitOps to validate if it is mature enough to be implemented in a corporate environment, to apply the best practices while operating with Kubernetes clusters. Apart from that, it is also intended to evaluate whether it reduces the time that a company's operations team spends investigating and resolving issues manually in the environment.

Keywords: GitOps, Kubernetes, DevOps, CI/CD, Cloud

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

Durante las últimas décadas se ha producido una gran revolución en la industria de las tecnologías de la información. El desarrollo de ámbitos como la computación en la nube no solo ha modificado la forma de diseñar y construir las aplicaciones, sino que, además, ha supuesto un cambio radical en la manera de administrar, aprovisionar y operar recursos y aplicaciones. Sumado a esto, cada vez más proyectos empezaron a adoptar metodologías ágiles durante las fases de planificación y desarrollo. Sin embargo, los equipos de operaciones aún seguían trabajando de manera tradicional y como consecuencia, dicha agilidad se veía lastrada, ya que no se conseguía llevar los cambios de las aplicaciones a producción a la misma velocidad a la que se desarrollaban nuevas características.

El concepto de DevOps surgió con el objetivo de poder mejorar la comunicación e integración entre los equipos de desarrolladores y operaciones, los cuales históricamente tenían objetivos diametralmente opuestos. Gracias a la filosofía DevOps, hoy en día se ha conseguido que ambas disciplinas trabajen de manera conjunta y eficiente, lo que reduce significativamente el tiempo de puesta en producción de los cambios en el código fuente. Este objetivo se ha conseguido gracias disponer de un alto grado de automatización en el área del desarrollo y construcción del software. Por contra, en otros ámbitos como la creación de infraestructura de aplicaciones y su aprovisionamiento aún no se ha conseguido alcanzar el mismo grado de madurez de la automatización en sus procesos.

El marco de trabajo GitOps, término acuñado por la empresa Weaveworks en 2017 (Weaveworks, 2017), surge con el objetivo de trasladar las mejores prácticas del desarrollo de aplicaciones a la creación y aprovisionamiento de la infraestructura en las mismas. Gracias a esto se consigue reducir la brecha de la automatización entre los distintos procesos de todo el ciclo de vida de la aplicación (Application Lifecycle Management, ALM) lo que permite disponer de aprovisionamientos y despliegues más repetibles y auditables. Como consecuencia, se consigue un mayor grado de automatización y como resultado una puesta en producción mucho más eficiente y mantenible en referencia a los cambios que los desarrolladores generan en las distintas aplicaciones.

1.1. Justificación del trabajo

La gestión de las operaciones en cualquier proyecto se compone de procesos muy sensibles y propensos a fallos si se ejecutan manualmente. La filosofía DevOps pone el foco en la colaboración entre equipos y en la automatización con el objetivo de conseguir reducir el tiempo que se tarda en lanzar nuevas características al eliminar la intervención manual en dichos procesos. Sin embargo, no solo basta con tener automatismos, ya que el conocimiento que el operador tenga del estado del sistema también es un factor crítico que ha de tenerse en cuenta. Este escollo se ha solventado gracias al uso de la Infraestructura como código (IaC, del inglés Infrastructure As Code), la cual utiliza lenguajes declarativos que permiten describir el estado del sistema, de manera que cualquier miembro del equipo pueda conocer los componentes que componen el sistema y sus interacciones.

Aun siendo un gran paso en la dirección correcta, esta aproximación se puede seguir evolucionando. El hecho de que el estado de la infraestructura quede reflejado de manera sencilla en un conjunto de ficheros, o manifiestos, no significa que este coincida con el que se encuentra en ejecución. Pueden existir una serie de factores externos que afecten al estado del sistema de manera que este cambie sin que los operadores sean conscientes de ello. Como consecuencia, se pueden generar derivas de la configuración y del estado que requieran la intervención del equipo de operaciones.

Debido a esto, y, aunque la filosofía DevOps aboga por la automatización, es necesario seguir mejorando los procesos relacionados con el aprovisionamiento y despliegue de infraestructura, ya que son puntos críticos que pueden generar grandes problemas a la hora de la puesta en producción y operación de aplicativos.

Por todo lo mencionado anteriormente, es fundamental poner solución a esta problemática a fin de conseguir que los procesos relacionados con los despliegues y aprovisionamientos sean lo más transparentes posibles y que el estado de estos sea siempre conocido.

1.2. Planteamiento del problema

Este trabajo de fin de Máster surge de la necesidad que tiene la empresa en la que me encuentro empleado de aplicar las mejores prácticas a fin de disponer de procesos eficientes para la gestión de clústeres de Kubernetes y de esta manera poder evitar problemas asociados a la configuración y el estado de estos.

Se ha detectado que aun disponiendo de automatizaciones y de herramientas no se dispone de un proceso que mitigue los problemas a la hora de operar con los clústeres de Kubernetes que se operan en la actualidad. Estas derivas pueden hacer que las nuevas versiones de las aplicaciones que se van a desplegar fallen o que los operadores no dispongan de la realidad sobre el estado actual del sistema. Como consecuencia, es necesario dedicar largas sesiones de depuración para conocer el origen del problema a fin de poder ponerle solución. Esta problemática impacta directamente en el equipo de operaciones, ya que sus miembros tienen que analizar la causa del problema y no pueden realizar otras tareas que aporten un mayor valor para la empresa o clientes.

Tras estudiar la problemática actual, se ha decidido poner en práctica el marco operacional GitOps, con el que se pretende optimizar el tiempo de los miembros del equipo de operaciones mediante la mitigación de las derivas de la configuración y el estado de la infraestructura y aplicaciones que ejecutan sobre los clústeres de Kubernetes. Además, permitirá seguir las mejores prácticas a la hora de operar los clústeres y se conseguirán beneficios como la auditabilidad, repetibilidad y eficiencia.

1.3. Estructura de la memoria

El documento se estructura de la siguiente manera:

1.3.1. Introducción

Este capítulo realiza una exposición de la problemática existente en el ámbito del despliegue y aprovisionamiento de la infraestructura de las aplicaciones en los clústeres de Kubernetes y las motivaciones que han dado lugar a llevar a cabo este trabajo.

1.3.2. Contexto y estado del arte

El objetivo de este capítulo es proporcionar al lector el contexto adecuado mediante el estudio del estado de DevOps y de las distintas metodologías y marcos de trabajo que están surgiendo para complementarlo, entre los que se encuentra GitOps. Adicionalmente, se mencionarán otros trabajos relacionados con la temática.

1.3.3. Objetivos y metodología de trabajo

En este capítulo se especifican los objetivos que se quieren conseguir al llevar a cabo este trabajo, su planificación y la metodología que se utilizará durante el desarrollo de este.

1.3.4. Desarrollo especifico de la contribución

Este capítulo cubre la investigación de las distintas tecnologías relacionadas con la puesta en marcha de GitOps, la selección de las tecnologías que se han decidido utilizar para realizar el trabajo, el diseño del flujo de trabajo utilizando los principios propuestos por GitOps y la implementación de este. Adicionalmente se mencionan los problemas encontrados durante el desarrollo.

1.3.5. Evaluación de resultados

Este capítulo cubrirá la evaluación del trabajo realizado a fin de estudiar la viabilidad de la implantación de GitOps en un entorno corporativo.

1.3.6. Conclusiones y trabajo futuro

En este capítulo se presentan las valoraciones técnicas y personales extraídas de la realización del trabajo de fin de máster. Por último, se exponen las posibles líneas futuras de investigación y trabajo.

2. Contexto y estado del arte

Este capítulo pretende proporcionar el contexto necesario para el Trabajo de Fin de Máster mediante la realización de un estudio sobre el estado de DevOps y las diferentes aproximaciones o metodologías que están surgiendo a su alrededor.

Adicionalmente, se llevará a cabo un análisis de los trabajos relacionados con el tema a fin de conocer su situación actual y madurez.

2.1. DevOps

La palabra DevOps es una combinación de los términos desarrollo (Development) y operaciones (Operations). (Gartner, 2019) define DevOps como un cambio organizacional que pretende acelerar la entrega de los productos y servicios mediante la mejora de aspectos como la colaboración entre los profesionales de desarrollo y operaciones y la automatización. Por otra parte, (Brown, 2015) define el término como *"DevOps is the union of people, process, and products to enable continuous delivery of value to our end users."* cuya traducción al español sería *"DevOps es la unión de personas, procesos y productos para permitir la entrega continua de valor a nuestros usuarios finales."*. Aunque el término no tenga una definición exacta se puede ver como diferentes fuentes coinciden en la importancia de DevOps como catalizador de la mejora de la eficiencia de los procesos de TI la satisfacción de los usuarios finales. Desde el punto de vista académico, múltiples autores también captan la misma esencia de la mejora de la comunicación y colaboración, aseguramiento de la calidad, integración y entrega continua y automatización (Jabbari et al., May 24, 2016).

El origen de la expresión DevOps se remonta al año 2009, durante la conferencia *"Velocity"* de O'Reilly. En dicha conferencia, John Allspaw y Paul Hammond exhibieron su trabajo *"10+ Deploys per-Day: Dev and Ops Cooperation at Flickr"* (Allspaw & Hammond, 2009). Unos meses después Patrick Dubois organizó la primera edición de la conferencia *"DevOpsDays"* que tuvo lugar en Bélgica.

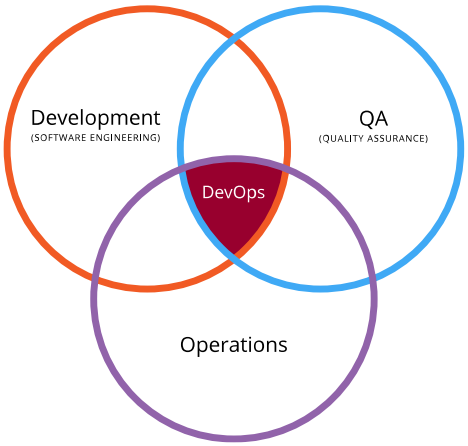
Cabe destacar que DevOps está muy ligado al desarrollo ágil y que es una de las causas del surgimiento de esta filosofía. Ambas permiten satisfacer las necesidades de las partes interesadas e incrementar la colaboración entre dos grupos con intereses distintos. Mientras que las metodologías ágiles abogan por la colaboración de los desarrolladores y la gente de

negocio, DevOps traslada esto a los equipos de desarrolladores y operaciones (Jabbari et al., May 24, 2016).

El objetivo de las metodologías ágiles, principalmente basadas en el manifiesto ágil, es el de que los equipos puedan mejorar la colaboración y adoptar cambios en los requisitos de manera que se pueda satisfacer las necesidades de los clientes de la mejor manera posible. Dichas metodologías o marcos de trabajo tales como Scrum o Extreme Programming (XP) permiten complementar DevOps ya que comparten características comunes, aunque las apliquen a equipos distintos. Además, DevOps también bebe de la cultura Lean ya que se pretende reducir o eliminar el desperdicio o lo que es lo mismo, trabajo que no aporta valor al producto (Buehring, 2021).

DevOps fue creado para poder solucionar las fricciones que tradicionalmente han existido entre los equipos de desarrolladores y operaciones y poner una solución al dilema que plantean ambos equipos (Buchanan, 2020). Mientras que los primeros buscan implementar cambios lo más rápido posible, los segundos abogan por la estabilidad y los sistemas libres de interrupciones. Esta situación crea fricciones ya que no existe un objetivo común por el que ambos equipos deban trabajar conjuntamente. Además, cabe destacar la gran importancia de la calidad del software ya que no solo basta con realizar entregas rápidas si no que el software debe ser fiable. Debido a esto DevOps suele verse como una intersección del desarrollo, operaciones y calidad (ver Figura 1).

Figura 1. *Diagrama Venn DevOps.*

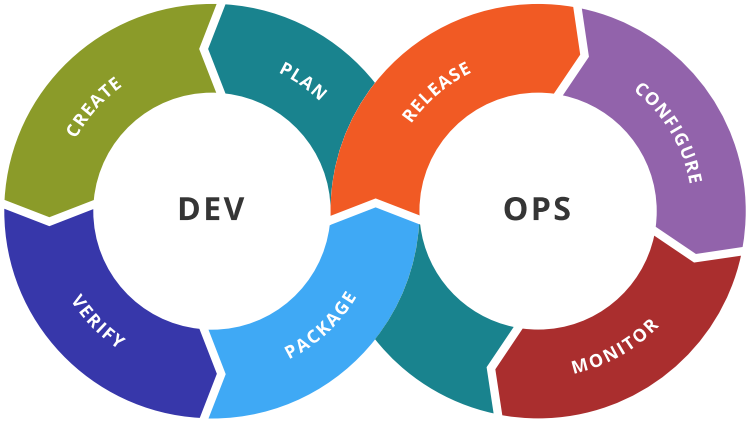


Fuente. Wikipedia.

2.1.1. El ciclo de vida de DevOps

El ciclo de vida de DevOps, representado con un símbolo infinito que simboliza la mejora continua e iterativa, pretende dividir el proceso en varias fases (ver Figura 2). La definición de dicho ciclo no está totalmente acotada por lo que dependiendo de la fuente el ciclo estará dividido en fases con distinto nombre. A pesar de esto, el significado de este siempre es el mismo.

Figura 2. *Ciclo de vida DevOps.*



Fuente. Wikipedia.

Según (Atlassian, 2021) estas fases son: planificar, compilar, integración y entrega continua, supervisión y alerta, operar y retroalimentación continua. Sin embargo, para (Microsoft, 2019) las fases son: planificar, desarrollar, entregar y operar. El nivel de granularidad en la definición de las fases difiere entre fuentes por lo que para definir estas se ha utilizado la definición de (Dhaduk, 2022) ya que realiza una división del ciclo en siete fases distintas.

2.1.1.1. Planificación (Plan)

Es la etapa en la que se lleva a cabo la definición de la hoja de ruta del proyecto, se identifican los requisitos y se recupera la retroalimentación del producto entregado en la iteración anterior.

2.1.1.2. Codificación (Code):

En esta fase es en la que el código es producido por parte de los miembros de los equipos de desarrollo.

2.1.1.3. Compilación/Construcción (Build):

En esta etapa es en la que se construye el código, es decir, se generan los binarios o los paquetes que estarán listos para ser desplegados en los entornos.

2.1.1.4. Prueba (Test):

Es la fase en la que se llevan a cabo los distintos tipos de pruebas. Entre ellas destacan las pruebas de aceptación, de integración, de rendimiento y de seguridad. Estas pruebas se suelen ejecutar en entornos muy parecidos a los de producción.

2.1.1.5. Lanzamiento (Release):

En esta etapa el software ha pasado las pruebas correspondientes y puede ser liberado. En este punto el software está listo para ser entregado al cliente.

2.1.1.6. Despliegue (Deploy):

Esta es la fase en la que el software es puesto en producción. En esta etapa se suele incluir la construcción del entorno de producción mediante infraestructura como código.

2.1.1.7. Operación (Operate):

En esta fase los clientes ya disponen del software y el equipo de operaciones estará encargado de configurar y operar el software.

2.1.1.8. Supervisión (Monitor):

Es la etapa en la que se recopilan datos del comportamiento del cliente, rendimiento del software, etc. Gracias a esta monitorización en la siguiente iteración se dispondrán de los datos necesarios para poder ajustar mejor la planificación y los objetivos.

2.1.2. Valores de DevOps

Debido al hecho de que las diferentes partes interesadas tienen un concepto diferente de lo que es DevOps existen un conjunto de valores comunes que permiten que dichas partes se puedan estar de acuerdo. Estos valores, abreviados comúnmente como CALMS (DeBoer, 2019), son los siguientes:

2.1.2.1. Cultura (Culture):

Pretende ajustar la cantidad de los procesos para que no existan dificultades en la comunicación y colaboración pero que a la vez no se conviertan en una carga para la productividad.

2.1.2.2. Automatización (Automation):

El objetivo es el de hacer progresar la automatización mejorando su flujo mediante el uso de la tecnología.

2.1.2.3. Simplificación (Lean):

Pretende eliminar cualquier desperdicio o trabajo que no aporte valor de manera que el esfuerzo siempre sirva para aumentar el valor del producto que se entrega.

2.1.2.4. Medición (Measurement):

No se puede mejorar si no se llevan a cabo mediciones que nos permitan conocer el estado en el que nos encontramos. Esto es un punto clave ya que DevOps aboga por la mejora continua. Debido a esto es necesario que se midan los procesos, las tecnologías y las personas.

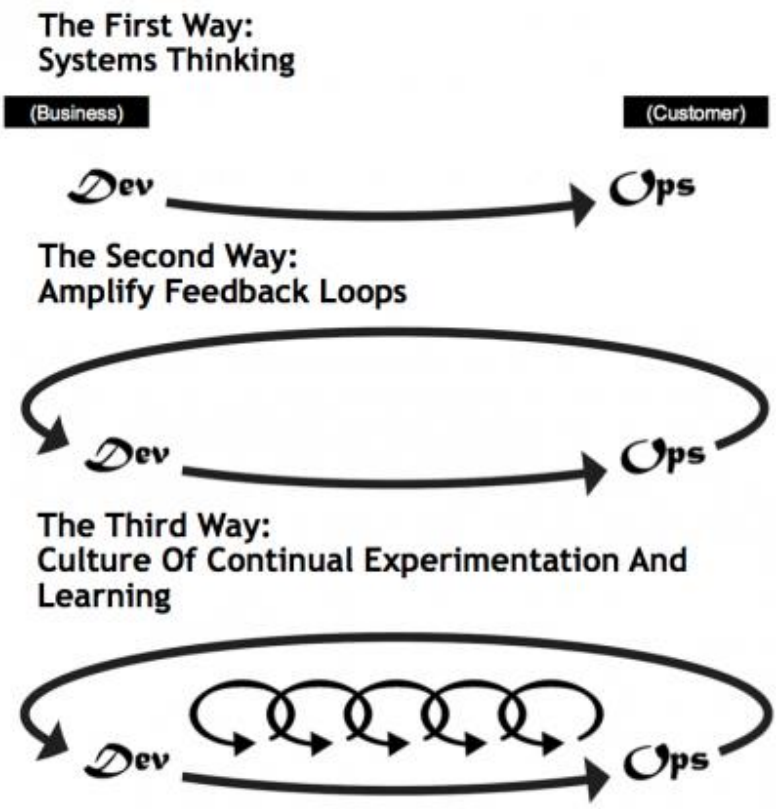
2.1.2.5. Intercambio (Sharing):

El objetivo de este valor es el de compartir el conocimiento entre las personas. Gracias al intercambio de ideas se consigue una mejora en la comunicación y en la colaboración y por extensión ayuda a mejorar la organización.

2.1.3. Principios de DevOps

Los principios de DevOps, también conocidos como las tres vías, son una serie de pasos que se pueden seguir para que la organización pueda adoptar y aplicar las prácticas de DevOps (Kim et al., 2021). Estas vías (ver Figura 3), presentadas en el libro *"The Phoenix Project: A novel about IT, DevOps and Helping your business Win"* de los autores Gene Kim, Kevin Behr y George Spafford, son el punto de partida del que surgen todos los patrones de DevOps ya que encuadran las prácticas y procesos de la disciplina (Kim, 2012).

Figura 3. *Tres vías de DevOps.*



Fuente. Kim, Gene. IT Revolution. The Three Ways: The Principles Underpinning DevOps.

2.1.3.1. Primera vía: Pensamiento de sistémico o de flujo

Esta vía intenta enfatizar el pensamiento sistémico o de flujo con el objetivo de optimizar el proceso de negocio, la agilidad y la confiabilidad. Se centra en eliminar restricciones del flujo de trabajo para que este pueda verse incrementado de principio a fin.

2.1.3.2. Segunda vía: Amplificar bucles de retroalimentación

El objetivo de la segunda vía es acortar los bucles de retroalimentación de manera que se puedan corregir los problemas detectados de manera ágil y continua.

2.1.3.3. Tercera vía: Cultura de la experimentación y el aprendizaje continuo

La tercera vía se centra la creación de una cultura que abogue por la experimentación continua y el aprendizaje del fracaso. Se debe entender que la mejor manera de conseguir el dominio sobre algún tema es a base de práctica y repetición.

2.1.4. Prácticas DevOps

Existen un gran número de prácticas DevOps que permiten soportar cada uno de los principios ya mencionados (Amazon Web Services, 2020). Gracias a estas prácticas se consigue entregar valor de manera mucho más rápida a los usuarios finales. Las principales son:

- **Integración continua**. Los desarrolladores son capaces de integrar código en un repositorio común múltiples veces al día y se ejecutan pruebas automatizadas que valida los cambios.
- **Entrega continua**. Consiste en que el software esté en un estado que permita entregar una nueva versión en cualquier momento al cliente.
- **Despliegue continuo**. Permite el despliegue automático del software en producción tras pasar las pruebas automatizadas de integración, seguridad, etc.
- **Pruebas continuas**. Consiste en la ejecución de todo tipo de pruebas de manera automática permitiendo validar el correcto funcionamiento del software, su robustez, seguridad de manera que se reduzcan los fallos en el entorno de producción.
- **Control de versiones**. Elemento clave que permite llevar a cabo un seguimiento de las revisiones de la evolución del código. Gracias a esta práctica se dispone del histórico de los ficheros, lo que permite llevar a cabo retrocesos en caso de haber introducido inestabilidad en el sistema tras desarrollar una nueva característica. Además, permite que los desarrolladores colaboren a la hora de llevar a cabo los desarrollos.
- **Infraestructura como código**. Permite describir el estado deseado de los sistemas de manera declarativa y tratar la infraestructura como si fuese código. Gracias a esto último se pueden aplicar técnicas de desarrollo como el control de versiones, integración continua y las pruebas automáticas. El principal beneficio de esta técnica es que se reduce el riesgo de error y se consigue implementar recursos de manera confiable, repetible y automatizada.

La mayoría de las prácticas mencionadas se llevan a cabo mediante el uso de herramientas. Estas herramientas se pueden integrar mediante APIs para generar una cadena que permita llevar a cabo todo el proceso de un extremo a otro de manera automática.

Gracias a la aplicación de estas prácticas se pueden obtener beneficios como la mejora en la frecuencia de despliegues, la reducción de los tiempos de recuperación ante fallos o una mejora la tasa de errores del software.

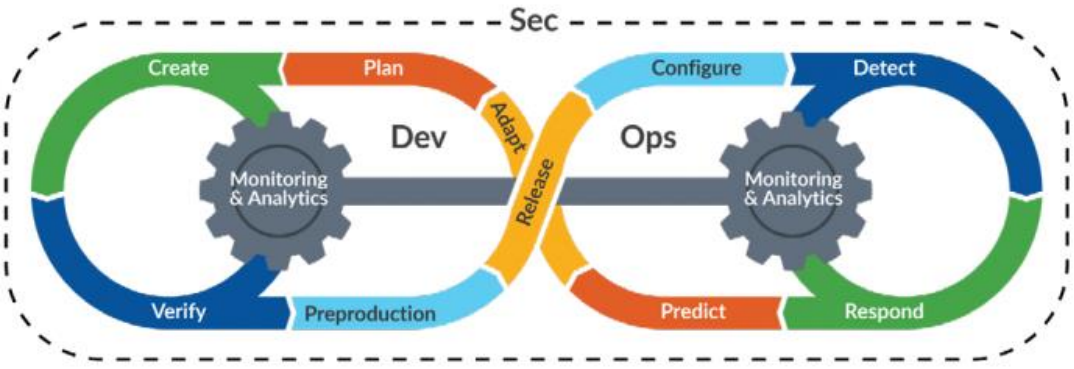
Cabe destacar que el *"State of DevOps 2021"* ha enfatizado la importancia de construir una plataforma propia que se ajuste a la organización y proporcione al resto de equipos características *"self-service"*. Esta plataforma, conocida como plataforma como producto, es gestionada por un equipo dedicado el cual actúa de habilitador para el resto. Gracias la filosofía *"self-service"* se consigue una mejor experiencia para los desarrolladores y reducción del trabajo manual del equipo de la plataforma. GitOps es un punto clave que permite operar de esta manera a fin de mejorar la eficiencia y reducir el trabajo manual de los equipos (Weaveworks, 2021).

2.2. Extensiones de DevOps

Durante estos años, gracias al gran éxito y adopción que la cultura de DevOps ha tenido, han surgido nuevos términos para designar culturas que tienen como objetivo fortalecer o incluir algún aspecto nuevo en el ciclo de DevOps. Estas extensiones de la cultura DevOps pretenden que aspectos como la seguridad, la inteligencia artificial y el análisis de datos se integren en el ciclo actual a fin de que obtengan la importancia que se merecen. Gracias a esto, hoy en día se está llevando a cabo la creación de un ecosistema de extensiones que pretenden ampliar la cultura DevOps y trasladar los beneficios mencionados con anterioridad a otros ámbitos de la informática.

2.2.1. DevOps Seguro (SecDevOps/DevSecOps)

Figura 4. *Integración de seguridad en DevOps.*



Fuente. Vera, José Manuel. Revolución SecDevOps.

Este paradigma ha ganado mucha fuerza en los últimos años ya que la seguridad es un punto fundamental para cualquier organización y en especial para aquellas que basan su negocio en la creación de software. Tradicionalmente, los departamentos de seguridad eran los encargados de auditar los sistemas y aplicaciones a fin de probar su robustez ante potenciales amenazas.

Este proceso de auditoría, en la mayoría de los casos, se llevaba a cabo de manera manual antes de que las aplicaciones fuesen desplegadas en entornos de producción o entregadas a los clientes. Además, una vez puesta en producción, la aplicación debía seguir siendo verificada por ingenieros de seguridad. El auge de internet y el surgimiento de las metodologías ágiles y de DevOps ha permitido que los procesos de entrega del software se optimicen y se aceleren hasta el punto de dejar obsoleto el modelo tradicional de auditoría que falla al intentar mantener el ritmo del resto de los procesos.

Este ha sido el detonante que ha hecho que la seguridad tenga que integrarse en el ciclo de DevOps a fin de poder entregar software de manera ágil que cumpla con los estándares y requisitos de seguridad exigidos.

Cabe destacar que existe un debate en cuanto a la nueva nomenclatura de esta metodología. Aunque los términos DevSecOps y SecDevOps puedan ser intercambiables en la mayoría de los contextos existen matices que hacen que la balanza se incline por el segundo (Peterson, 2020). La principal diferencia entre ambos términos radica en el matiz de que el SecDevOps pone la seguridad ante todo ya que no basta con que únicamente se integre en cada etapa del ciclo de vida del desarrollo del software. Lo que se pretende conseguir es promover las prácticas de seguridad y que la responsabilidad de dicho ámbito sea compartida entre todos los miembros del equipo.

Algunos de los principios que promueve SecDevOps son:

- **Responsabilidad**. Los equipos comparten por igual la responsabilidad en el ámbito de la seguridad.
- **Políticas de Seguridad bien definidas**. Las políticas de seguridad están correctamente definidas a nivel empresarial.
- **Seguridad aplicada a todo el ciclo**. La seguridad debe integrarse en todas las etapas del ciclo DevOps, incluyendo la de planificación.

- **Automatización**. La automatización de los procedimientos de seguridad es fundamental.

En SecDevOps los valores CAMS se analizan desde el punto de vista de la seguridad y su relación con la misma. (Koskinen, 2019) propone una serie de consejos prácticos a fin de poder aplicar la seguridad a dichos valores y de esta manera poder comprender la metodología de manera adecuada.

Tabla 1. *Consejos para la construcción de seguridad en el núcleo DevOps.*

Valores DevOps	Consejo práctico
Cultura	Desarrollar una cultura orientada a la seguridad con una fuerte conciencia de en este ámbito.
	Crear una cultura con sentido de responsabilidades compartidas, en la que los equipos de DevOps y seguridad trabajen juntos por un objetivo común.
	Priorizar la corrección de defectos de seguridad.
Automatización	Automatizar la seguridad tanto como sea posible, asegurándose de que las herramientas elegidas sean aplicables al entorno y la tecnología.
	Garantizar la seguridad de los pipelines.
	Hay que reconocer que la automatización mejora las actividades manuales de seguridad, no las reemplaza
Medición	Desarrollar métricas de seguridad.
	Medir y monitorizar la seguridad constantemente
	Proporcione retroalimentación a los desarrolladores y operadores sobre los problemas de seguridad.
Intercambio	Compartir conocimientos sobre principios de seguridad

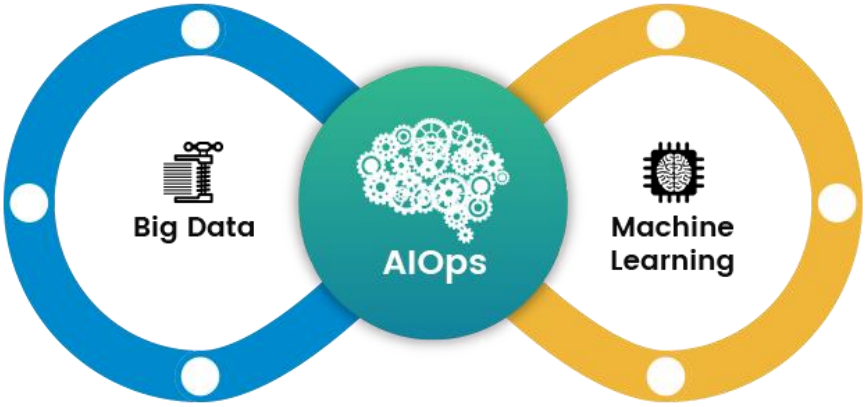
	Desarrollar la gestión de incidentes e involucrar a los desarrolladores en la solución de problemas de seguridad.

Fuente. Koskinen, Anna (2019). DevSecOps: Building security into the core of DevOps.

La implantación de una cultura SecDevOps puede aportar una gran cantidad de beneficios a las organizaciones, ya que la seguridad no es un requisito opcional y debe tenerse en cuenta desde la concepción del software si se espera poder entregar software seguro a los clientes.

2.2.2. AIOps

Figura 5. *Representación AIOps.*



Fuente. Alisha Fathima. Bootmetric. What is AIOps?

(Gartner, 2021) define AIOps (Artificial Intelligence Operations) como la automatización de los procesos de operaciones de TI, incluyendo la correlación de eventos, detección de anomalías y determinación de la causalidad, mediante la combinación de técnicas de Big Data y el aprendizaje automático. El objetivo principal es el de convertir los datos generados por las plataformas en información relevante.

La complejidad y sofisticación de los entornos de TI se ha incrementado de manera abrumadora en los últimos años. Esto, sumado al cambio de arquitectura de las aplicaciones hacia los microservicios y contenedores hace que la cantidad de componentes que el equipo de operaciones debe organizar y manejar sea inabarcable (Singh, 2019).

AIOps surge para intentar aprovechar el análisis de datos, Big data y aprendizaje automático a fin de poder atender las demandas de una infraestructura moderna y ágil. El uso de la inteligencia artificial facilita una toma de decisiones más rápida e informada.

Los casos de uso más habituales para AIOps son los siguientes:

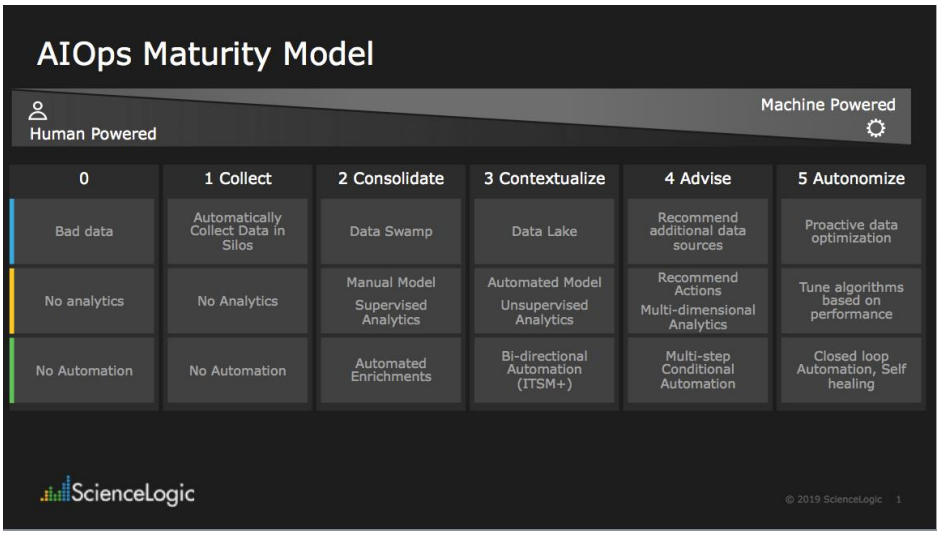
- **Detección de anomalías**. Permite ejecutar acciones correctivas tras la detección de la anomalía en los datos. Además, facilita la identificación de falsos positivos.
- **Análisis causal**. Facilita la realización del análisis del origen del problema para llevar a cabo las correcciones y evitar futuras ocurrencias.
- **Predicción**. Permite la predicción automática de sucesos que puedan impactar a la infraestructura o las operaciones como puede ser el incremente del tráfico de los usuarios.

AIOps complementa a DevOps ya que ambos comparten un objetivo común, incrementar la eficiencia de las organizaciones para que sean más productivas. Gracias a él los equipos y sus prácticas pueden ser más efectivos gracias a los datos que se generan. La observabilidad de los sistemas juega un papel clave en el desarrollo de esta metodología ya que se necesitan suficientes datos de calidad para lograr la automatización deseada. Además de lo anterior, AIOps implica un giro en la creación y tratamiento de datos de la organización ya que una estrategia de recolección y manejo de datos holística permitirá aprovechar todo el potencial que esta metodología puede proporcionar (Matthews, 2019).

Las herramientas de AIOps tienen como objetivo reducir el ruido de las alertas mediante la correlación alertas y de incidentes relacionados, la recolección de series temporales de datos, la creación de modelos de aprendizaje automático para agregarlos y recolección automática de telemetría.

A fin de ayudar a los clientes a comprender el estado de su organización en cuanto a AIOps se refiere, la empresa ScienceLogic ha creado un modelo de madurez basado en cinco etapas (ver Figura 6).

Figura 6. *Niveles de madurez de AIOps.*



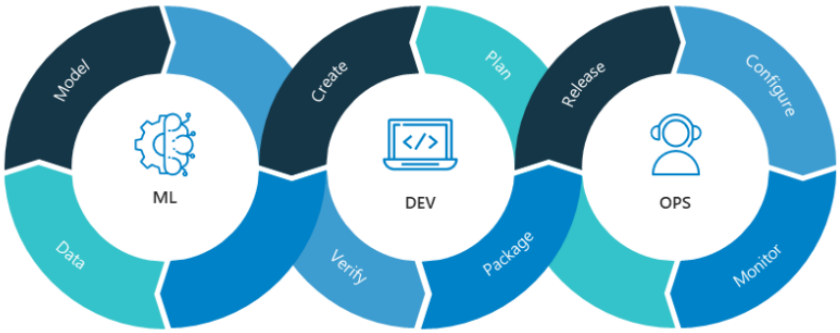
Fuente. ScienceLogic.

Actualmente la mayoría de las empresas se encuentran en los niveles de madurez uno y dos ya que recolectan datos, pero no existe una correlación clara entre ellos o están empezando a tener modelos manuales y pequeñas automatizaciones. En los niveles tres y cuatro se encuentran empresas pocas empresas y muy innovadoras. Estas empresas utilizan algoritmos no supervisados y de análisis multidimensional, además, disponen de un alto grado de automatización. Hoy en día ninguna empresa ha conseguido alcanzar el nivel cinco del modelo de madurez completamente (Branscombe, 2019).

Los beneficios de AIOps son bastante claros ya que permite reducir el tiempo de detección y solución de problemas lo que mejora la disponibilidad y fiabilidad del sistema. Además, a largo plazo permite reducir la tasa de incidentes y aumentar el tiempo medio entre fallos. AIOps facilitará la mejora de procesos de la empresa gracias al uso de la inteligencia artificial y los datos permitiendo que los ingenieros DevOps se dediquen a ser más productivos (ScienceLogic, 2019).

2.2.3. MLOps

Figura 7. *Ciclo de MLOps.*



Fuente. Merritt, Rick. Nvidia Blog. What is MLOps?

El creciente uso del aprendizaje automático y la inteligencia de negocio en los últimos años ha hecho que muchas empresas opten por crear productos, para uso interno o externo, que utilizan estas técnicas con el objetivo de beneficiarse de sus bondades. MLOps o Machine Learning Operations es un término que hace referencia al conjunto de las mejores prácticas que las empresas puedan aplicar para ejecutar procesos de inteligencia artificial de manera exitosa (Merritt, 2020). El objetivo principal de este marco de trabajo se centra en la optimización de los procesos de aprendizaje automático, su despliegue, mantenimiento y monitorización. MLOps consigue estandarizar y agilizar la entrega continua de modelos de aprendizaje automático para los entornos de producción.

Gracias a este enfoque se consiguen crear soluciones de inteligencia artificial y aprendizaje automático de calidad ya que los ingenieros DevOps colaboran estrechamente con los científicos de datos y los ingenieros de inteligencia artificial lo que se traduce en una mejor producción de modelos y más automatización.

La creación de modelos de aprendizaje automático es una tarea ardua ya que el ciclo de vida del aprendizaje automático consta de una gran cantidad de componentes complejos que deben tenerse en cuenta. Además, también requiere la colaboración de múltiples equipos como los de ingeniería de datos, ciencia de datos y equipo de operaciones. Todo esto provoca que se requiera una perfecta sincroniza entre los procesos y que la iteración y mejora continua durante este ciclo de vida sean fundamentales (Databricks, 2020). Entre los beneficios de utilizar MLOps destacan:

- **Eficiencia**. Acelera y facilita el desarrollo, despliegue y verificación de modelos de aprendizaje automático. Además, esto permite mejorar la calidad de estos.
- **Escalabilidad**. Permite la supervisión y administración de los modelos producidos, su entrega continua, y administración. Aporta reproducibilidad gracias a los pipelines de aprendizaje automático y acelera la velocidad de liberación.
- **Reducción de riesgos**. Proporciona transparencia y una respuesta más rápida a la hora de realizar la verificación de los modelos producidos. Además, facilita el cumplimiento de las políticas de la organización.

Es muy común que los términos AIOps y MLOps sean usados indistintamente lo cual no es un uso adecuado. Cabe destacar que el objetivo de ambos es el mismo, mejorar la eficiencia de los procesos de las organizaciones, sin embargo, el ámbito de ambos es totalmente distinto.

Mientras que AIOps pretende automatizar las operaciones del sistema utilizando técnicas de aprendizaje automático y big data, MLOps pretende estandarizar el proceso de creación y despliegue de los sistemas de aprendizaje automático además de mejorar la colaboración entre equipos y proporcionar información relevante a las partes interesadas (Sharma, 2021) (ver Tabla 2).

Tabla 2. *Diferencias entre MLOps y AIOps.*

MLOps	AIOps
Estandariza el proceso de desarrollo de los sistemas de aprendizaje automático.	Automatiza las operaciones y los sistemas de TI.
Aumenta la eficiencia y productividad del equipo.	Automatiza el análisis y resolución del origen del problema.
Agiliza la colaboración entre diferentes equipos.	Procesa y gestiona una gran cantidad de datos de forma eficaz y eficiente.
Es una parte fundamental del despliegue a gran escala de manera repetible de la inteligencia artificial y la ciencia de datos.	Aprovecha las tecnologías de la IA para resolver desafíos de TI.

• Consumo de datos de múltiples fuentes. • Control de código fuente. • Servicios de despliegue y pruebas de seguimiento del modelo usando metadatos. • Automatización de experimentos de aprendizaje automático. • Mitigación de riesgos y sesgos en la validación del modelo.	• Monitorización de aplicaciones y automatización de procesos manuales y repetitivos. • Detección de anomalías. • Mantenimiento predictivo. • Gestión de incidentes.

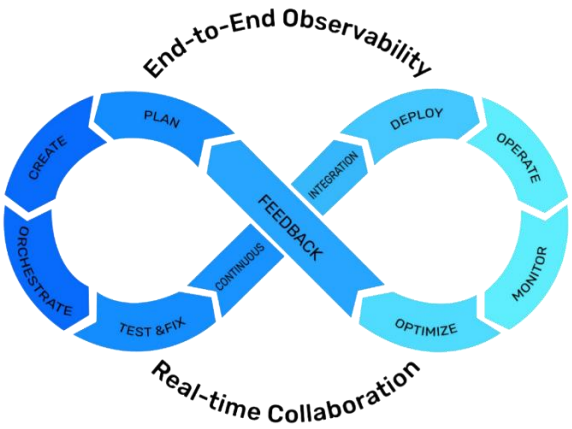
Fuente. Sharma. Natasha. Neptune blog. MLOps vs AIOps - What's the difference?

Aunque existan similitudes entre los proyectos de DevOps y MLOps cabe destacar que no es conveniente aplicar las prácticas de DevOps sin adaptar a los procesos de aprendizaje automático ya que pueden obtenerse resultados no satisfactorios. Es necesario involucrar a expertos en dicho ámbito para conseguir un modelo adecuado (Webster, 2021).

MLOps permite aplicar los principios de DevOps y aplicarlos al área del aprendizaje automático para conseguir producir modelos mejores y reducir el tiempo de despliegue de dichos modelos en los entornos de producción. Gracias a ambos se consigue una mayor calidad del software y una mayor satisfacción de los clientes.

2.2.4. DataOps

Figura 8. *Ciclo de vida de DataOps.*



Fuente. DataOps unleashed. What is DataOps?

DataOps (Data Operations) surge con el objetivo de que las organizaciones maximicen el valor de los datos que poseen. Hoy en día la gran mayoría de las empresas recolectan datos de alguna manera. Sin embargo, no son capaces de usar dichos datos para generar conocimiento

que les permita obtener valor comercial. El creciente volumen de datos que se están recolectando año tras año está complicando la gestión y análisis de estos y estos procesos se han vuelto procesos críticos que pueden generar cuellos de botella. DataOps pretende promover prácticas que mejoren la eficiencia y agilidad de los procesos que componen los pipelines de los datos desde su recolección hasta la entrega (IBM, 2019).

Esta mejora de los procesos pone el foco en acelerar la recopilación, procesamiento y el análisis de datos. Gracias a la optimización de estos procesos, la empresa podrá tomar mejores decisiones en todos los ámbitos lo que se puede traducir en la mejora de procesos internos o la creación nuevos productos o características que permitan conseguir una ventaja competitiva respecto a la competencia.

Uno de los puntos clave de esta metodología es el de reducir la brecha entre datos y conocimiento. Hoy en día la mayoría de las empresas disponen de ingentes cantidades de datos que pueden ser un arma muy poderosa a la hora de tomar decisiones si consiguen convertirse en conocimiento (Hemo, 2020).

DataOps surge de la combinación de DevOps, metodologías ágiles, *lean manufacturing*, y el control estadístico de procesos (Statistical Process Control - SPC). El enfoque ágil permite gobernar el desarrollo analítico, DevOps se encarga de la optimización de la verificación del código, construcción y entrega de los análisis y, por último, el control estadístico de procesos organiza, monitoriza y valida el pipeline de los datos (DataKitchen, 2022).

Cabe destacar que el control estadístico de procesos permite asegurar que la variación estadística se encuentre en rangos aceptables. Gracias a esto se consigue que los datos puedan ser verificados mientras fluyen a través del pipeline, lo que permite el envío de alertas al equipo correspondiente en el caso de que estos datos contengan alguna anomalía (ODSC-Open Data Science, 2019).

Al igual que DevOps, derivados o las metodologías ágiles, DataOps dispone de un manifiesto que pretende definir los principios de esta disciplina (TheDataOpsManifesto, 2022). Estos principios son los siguientes:

- Satisfacer continuamente al cliente
- Valorar las soluciones de analítica eficientes
- Abrazar el cambio

- Es un deporte de equipo
- Interacciones diarias
- Auto organización
- Reducir el heroísmo
- Reflexión
- La analítica es código
- Organización.
- Hazlo reproducible.
- Entornos desechables
- Sencillez.
- Analizar es producir
- La calidad es primordial
- Supervisión de la calidad y rendimiento
- Reutilización
- Mejorar la duración de los ciclos

Gracias a DataOps se consigue poner el en punto de mira de las organizaciones los datos y su tratamiento. El tratamiento de las grandes cantidades de datos que se están generando actualmente es un problema que DataOps pretende solucionar mediante la aplicación de enfoques ágiles y DevOps.

2.2.5. BizDevOps

Figura 9. *Ciclo de vida de BizDevOps.*

Fuente. Putano, Ben. Stackify. A quick guide to BizDevOps.

BizDevOps (Putano, 2017) tiene como objetivo mejorar la colaboración entre los equipos de negocio y los de desarrollo y operaciones. Es necesario que estos equipos compartan ideas y participen en bucles de retroalimentación durante todo el ciclo de vida del proyecto. La mejora de esta colaboración tiene como beneficio una mejora tanto en los resultados y la eficiencia del producto como en la conclusión exitosa del proyecto desde el punto de vista empresarial (AppDynamics, 2022).

Uno de los principales motivos por los que este enfoque es necesario es la alta tasa de fracaso en los proyectos de TI. El motivo de los fracasos de estos proyectos, en su mayoría, viene dado por una falta de alineación entre los equipos de negocio, desarrollo y operaciones (Ismail, 2018). Esto puede deberse a una mala definición de los requisitos o a una falta de comunicación entre los desarrolladores y las partes interesadas.

Una de las barreras existentes hoy en día y que dificulta la comunicación entre las partes interesadas, el equipo de negocio y los desarrolladores es que estos no disponen de un idioma común. Por norma general, los dos primeros no entienden el código de la aplicación evitando que pueda existir una comunicación efectiva entre los equipos.

Para mitigar este problema existen herramientas de programación sin código (*low code*) que permiten utilizar técnicas de desarrollo visuales para definir las interfaces de usuario, modelo de datos o lógica de negocio de una aplicación (Mendix, 2022). Gracias a este tipo de herramientas se consigue un nivel de abstracción que permite que los equipos se comuniquen de manera eficaz lo que además promueve el intercambio de ideas, la experimentación y el prototipado.

Adicionalmente, el uso de documentación viviente se hace fundamental. La documentación viviente, escrita por el equipo de negocio, pretende plasmar sin ambigüedades la intención comercial que se tiene (Geertsema, 2021). Esta documentación tiene que ser comprensible por el equipo técnico por lo que es recomendable el uso de lenguajes estándares como el de Modelo y Notación de Procesos de Negocio (del inglés *Business Model Process and Notation BMPN*). Los ingenieros utilizarán dicha documentación para generar código por lo que esté estará siempre alineado con los intereses comerciales.

Algunas de las ventajas que proporciona la aplicación de BizDevOps son el fomento de la comunicación y colaboración entre negocio y desarrollo/operaciones, la creación de entornos

que facilitan y promueven la innovación, una mejora de la eficiencia que permite reducir costes e incrementar el retorno de la inversión y una alineación casi perfecta entre los ámbitos de negocio y técnico otorgando a la empresa una gran ventaja competitiva.

2.2.6. GitOps

(Weaveworks, 2018) define GitOps como un modelo de trabajo para Kubernetes y entornos nativos de la nube. Este modelo establece un conjunto de buenas prácticas con el objetivo de fusionar el despliegue, la gestión y la monitorización de clústeres de Kubernetes y aplicaciones basadas en contenedores.

Este modelo busca conseguir una experiencia centrada en el desarrollo para la gestión de aplicaciones. Para conseguir esto traslada varias de las prácticas del desarrollo de software, tales como los flujos de trabajos con Git, a la creación y aprovisionamiento de la infraestructura.

Cabe destacar que GitOps puede aplicarse en infraestructura que no se basa en Kubernetes. Sin embargo, dicha infraestructura tendrá que cumplir los siguientes requisitos:

- Debe ser observable.
- Puede describirse de forma declarativa.
- Dispone de herramientas de infraestructura como código.

Este modelo de trabajo se pensó para trabajar en entornos de Kubernetes por lo que aprovecha muchas de sus propiedades. Entre ellas destacan:

- **Automatización**. Las actualizaciones proporcionan un mecanismo para automatizar la aplicación de cambios de manera correcta.
- **Convergencia**. Kubernetes intentará actualizar el estado hasta que lo consiga.
- **Idempotencia**. Múltiples aplicaciones del mismo estado tendrán el mismo resultado.
- **Determinismo**. El estado del clúster depende únicamente del estado deseado.

El fundamento de GitOps es el de disponer de un repositorio de control de versiones, normalmente Git, que contenga la descripción en lenguaje declarativo del estado deseado de la infraestructura en los diferentes entornos y un proceso automático que sea el encargado de que la infraestructura desplegada coincide con el estado descrito en el repositorio (Beetz et al., 2022).

(Weaveworks, 2018) proporciona una descripción pormenorizada de GitOps a fin de entender el proceso completo.

Inicialmente se debe describir el estado deseado del sistema utilizando un lenguaje declarativo para cada entorno. Una vez descrito el estado, este debe almacenarse en un sistema de control de versiones que actúa como fuente de verdad. Los commits sobre el repositorio que contiene el estado son el mecanismo utilizado para generar cambios en el mismo.

Si el estado deseado y el observado son iguales se dirá que son convergentes mientras que si son diferentes son divergentes.

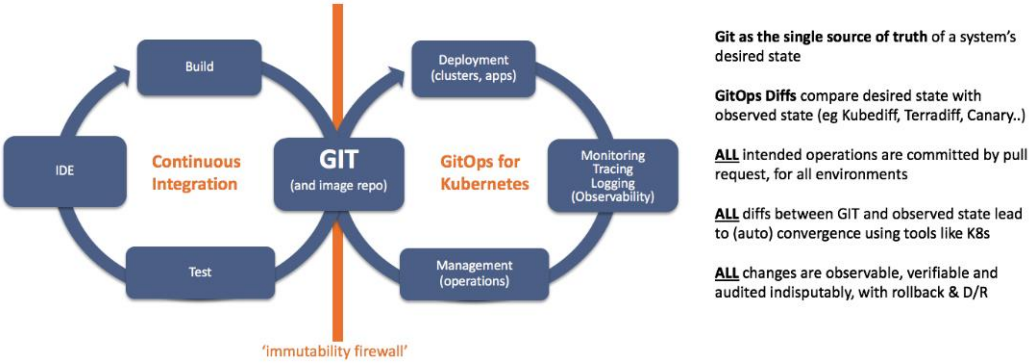
En el caso de que el estado observado y deseado sean divergentes existe un mecanismo automático (Kubernetes) que intentará que el primero alcance el segundo. Este mecanismo se dispara al añadir un nuevo cambio en Git.

Tras un intervalo de tiempo se podrá notificar sobre el resultado de la operación y las diferencias entre ambos estados en el caso de ser divergentes.

Cabe destacar que los commits permiten generar actualizaciones idempotentes y verificables. Gracias a la transaccionalidad de Git se consigue una actualización del estado deseado que deriva en un flujo de despliegue continuo en el que la actualización del clúster es atómica. Además, el uso de control de versiones facilita el mecanismo de regresión hacia un estado anterior.

Por último, se debe señalar que la convergencia de los estados es eventual ya que el proceso puede tardar tiempo en aplicar los cambios para alcanzar el objetivo. La Figura 10 muestra el modelo operativo que sigue GitOps.

Figura 10. *Modelo operativo de GitOps.*



Fuente. Weaveworks. What is GitOps?

Como puede observarse existen 2 artefactos clave en este flujo que son compartidos por ambos ciclos: el repositorio de imágenes y Git.

Entre los beneficios de GitOps destacan:

- **Despliegues más rápidos y frecuentes**. Facilita el despliegue continuo y no requiere de herramientas adicionales.
- **Recuperación de errores más rápida y sencilla**. Gracias a Git y su histórico se pueden recuperar versiones anteriores.
- **Fácil gestión de credenciales**. El entorno necesita acceso al repositorio de Git y de imágenes, los desarrolladores no necesitan ningún tipo de credencial.
- **Despliegues autodocumentados**. La descripción del despliegue está siempre disponible en el repositorio de Git por lo que se puede ver tanto el estado actual como estados pasados.
- **Conocimiento compartido**. El equipo puede ver la evolución de los despliegues y participar en el proceso de *Pull Request/Merge Request* por lo que es más fácil detectar fallos.

GitOps requiere al menos dos repositorios, el repositorio de aplicaciones y el repositorio de configuración del entorno. El primero contiene el código fuente de las distintas aplicaciones y los manifiestos de despliegue de estas, mientras que el segundo contiene todos los manifiestos de configuración de la infraestructura deseada que describen las aplicaciones y servicios que se deben ejecutar.

Existen dos maneras de implementar la estrategia de despliegue con GitOps. Estas estrategias indican como se dispara el proceso de despliegue dentro del clúster (Beetz et al., 2022).

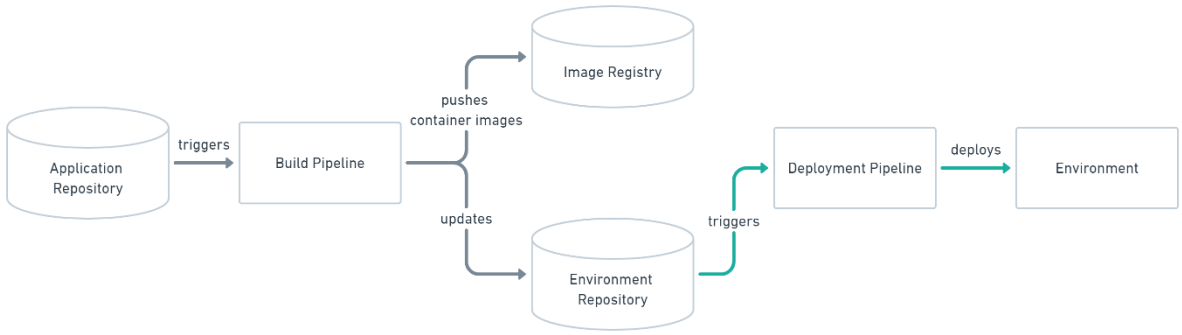
2.2.6.1. Despliegues basados en push

En este tipo de despliegue el código de la aplicación y los ficheros de manifiesto de YAML conviven en el repositorio de aplicación. Cada vez que el código se actualiza se dispara un pipeline que lleva a cabo el compilado, la generación de las imágenes y su envío al repositorio y por último lleva a cabo la actualización del repositorio de configuración con los nuevos manifiestos.

A continuación, la actualización en el repositorio de configuración dispara un pipeline de despliegue que aplica los ficheros de manifiesto en el clúster. Una de las desventajas de este método es que requiere que el pipeline de despliegue disponga de credenciales para actualizar la infraestructura. Además, con este modelo no se pueden detectar las desviaciones en el estado deseado por lo que no se intentan corregir automáticamente.

Con este modelo solo se obtiene como beneficio una mejora de la automatización ya que esta manera de trabajar es una pequeña automatización de como aplicaría los cambios un operador que trabajase directamente sobre el clúster a través de su interfaz de línea de comandos. La Figura 11 muestra un diagrama de este proceso.

Figura 11. *Despliegue basado en push.*



Fuente. Beetz et al. GitOps tech. GitOps.

2.2.6.2. Despliegues basados en pull

En este modelo se introduce el concepto de operador. Este operador es el encargado de comparar el estado deseado del repositorio con el estado actual del clúster. En el caso de que los estados sean diferentes el operador actualizará la infraestructura para llevar el estado

observado al estado deseado. El estado que observa pueden ser los ficheros de manifiestos o las nuevas versiones de las imágenes que se actualizan en el repositorio.

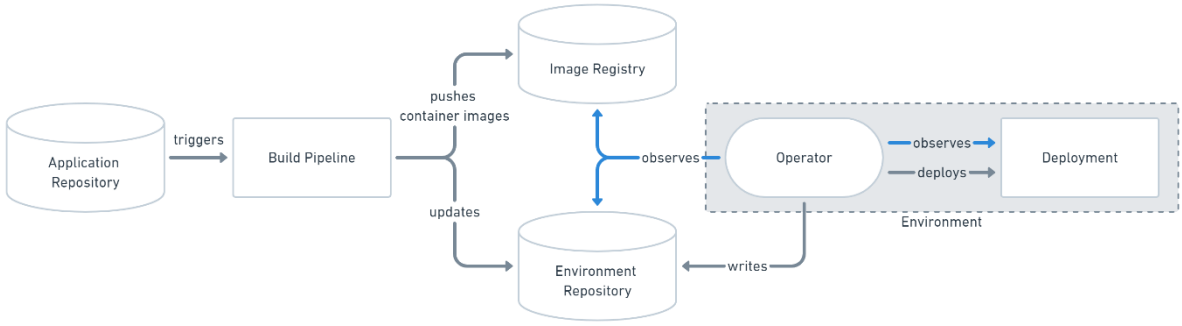
Adicionalmente, con este modelo se consiguen corregir las temidas derivas de la configuración que pueden ocurrir en el día a día ya que al detectar un cambio en el estado observado siempre intentara volver a desplegar el estado descrito en el repositorio. En el caso de no poder aplicar el estado se proporcionan mecanismos de notificación que permiten avisar a los operadores de posibles problemas.

Cabe destacar que el operador debe residir en el mismo clúster en el que la aplicación será desplegada. Gracias a esto se consigue evitar tener que proporcionar credenciales a servicios externos para desplegar en el entorno. El mecanismo de seguridad puede basarse en los que el control de acceso por roles proporcionado por Kubernetes.

Por último, es recomendable disponer de un mecanismo que monitorice el buen funcionamiento del operador encargado de hacer que los estados converjan.

La muestra este tipo de modelo.

Figura 12. *Despliegue basado en pull.*



Fuente. Beetz et al. GitOps tech. GitOps.

Los modelos mencionados anteriormente únicamente exponen el flujo de trabajo para una única aplicación, sin embargo, GitOps puede escalar para funcionar cuando se disponen de múltiples entornos o aplicaciones. Esta administración puede llevarse a cabo mediante el uso de ramas separadas para los entornos y configurando el operador del entorno para que únicamente observe su rama.

Una vez explicado el funcionamiento de GitOps se podría pensar que su relación con DevOps viene únicamente dada por el nombre. Mientras que DevOps aboga por un cambio cultural,

GitOps se centra mucho más en la pila tecnológica para conseguir mejorar las operaciones y despliegue de infraestructura. Aunque ambas técnicas parecen muy diferentes comparten los mismos principios (Beetz & Harrer, 2021).

Los aspectos que son parte de DevOps y no de GitOps son principalmente el enfoque iterativo y la fuerte colaboración. Sin embargo, aunque GitOps no requiera el trabajo iterativo de manera estricta el despliegue de pequeños incrementos en los diferentes entornos sigue este patrón ya que facilita la labor. Otro de los puntos a destacar es el carácter de autoservicio de GitOps, que se alinea completamente con el mismo aspecto de DevOps. Además, el uso de mecanismos de integración y despliegue continuos hace que compartan principios como la automatización.

Respecto a la monitorización, ambos persiguen el mismo objetivo, aunque GitOps pretende extender esta supervisión a la infraestructura ya que DevOps se promulga principalmente la monitorización de las aplicaciones.

El principal aporte de GitOps a DevOps es la evolución de la Infraestructura como Código. DevOps promueve el uso de este tipo de mecanismos sin entrar en detalle mientras que GitOps define un marco de mejores prácticas sobre cómo debe aplicarse.

GitOps pretende extender DevOps, no reemplazarlo. Hay que verlo como un marco operacional que permite mejorar el despliegue de aplicaciones basadas en el cloud en entornos altamente observables y automatizables como un clúster de Kubernetes. Gracias a esto se conseguirá optimizar el trabajo del equipo de operaciones y se evitarán una gran cantidad de problemas ya que el sistema será capaz de autogestionarse para que el estado interno sea siempre el deseado.

2.3. Trabajos relacionados

Existen una gran cantidad de trabajos relacionados que tratan el tema del despliegue de infraestructura en Kubernetes usando GitOps tanto desde un punto de vista teórico como práctico. Algunos de los más destacables e influyentes son los siguientes:

- **(Weaveworks, 2022)**. Tras acuñar el término de GitOps, ha sido uno de los principales precursores del movimiento a nivel empresarial. Su principal aporte ha sido realizado mediante la generación de una gran cantidad de documentación que clarifica e ilustra el funcionamiento de este.

- **(Gitlab, 2022)**. Esta empresa no solo ofrece documentación escrita, sino que además proporciona una gran cantidad de documentos audiovisuales que contienen tanto explicaciones como ejemplos de aplicación.
- **Autores como (Beetz et al., 2022)**. Estos autores han intentado tratar el tema desde un punto de vista académico. Este tipo de autores, que ha generado publicaciones en revistas de alto impacto científico, han tenido como principal objetivo conseguir generar una definición clara del término, estudiar su evolución y comparar sus principios con los de otros movimientos como el de DevOps.

2.4. Conclusiones del estado del arte

A pesar de su reciente creación, GitOps ha sido ampliamente estudiado en la industria. La mayoría de los autores coinciden en que facilita que los equipos de operaciones puedan incrementar su eficiencia hasta el punto de poder seguir el ritmo de los equipos de desarrollo. Además, ponen en gran valor los principios que aporta su implementación ya que permite trasladar de manera concreta los principios de DevOps a un entorno nativo en la nube.

3. Objetivos y metodología de trabajo

Este apartado pretende describir los objetivos que se quieren alcanzar durante la realización del Trabajo de Fin de Máster. Estos objetivos están categorizados en dos grupos diferenciados: generales y específicos. Adicionalmente, se presentará la metodología que se pretende seguir para conseguir dichos objetivos durante la duración de este trabajo.

3.1. Objetivo general

Implementar un flujo de trabajo de GitOps que permita desplegar y aprovisionar infraestructura de manera automática en un clúster de Kubernetes sin que se tenga que aplicar dicha configuración manualmente sobre el mismo. Gracias a este flujo de trabajo se pretende conseguir una administración más eficiente de las configuraciones de los clústeres de Kubernetes.

3.2. Objetivos específicos

Los objetivos específicos son los siguientes:

- **Explorar el contexto de DevOps** y las diferentes metodologías que han surgido a raíz de este.
- **Investigar el marco de trabajo GitOps** como una de las buenas prácticas para la gestión de clústeres de Kubernetes.
- **Identificar las herramientas** que se pueden usar para llevar a cabo la implementación de dicho marco.
- **Implementar un flujo de trabajo de GitOps** sobre un clúster de Kubernetes utilizando una aplicación desarrollada como *"cloud-native"*.
- **Analizar la viabilidad del marco de trabajo** para implementarlo en un entorno corporativo.

3.3. Metodología del trabajo

Los objetivos anteriores se han dividido en una serie de tareas que tendrán que llevarse a cabo para conseguir finalizar el Trabajo de Fin de Máster. Estas tareas son las siguientes:

- Estudio del contexto de DevOps.
- Análisis de las metodologías surgidas a raíz de DevOps.

- Identificación de los requisitos necesarios para implementar GitOps.
- Análisis de las principales herramientas y tecnologías del mercado para GitOps.
- Búsqueda de una aplicación de demostración que se usará en el flujo de trabajo.
- Elaboración del diseño del flujo de trabajo en GitOps.
- Implementación del flujo de trabajo diseñado.
- Análisis de la viabilidad de la implementación del marco de trabajo en un entorno corporativo.
- Redacción de la memoria del trabajo.
- Creación de la presentación para la exposición.

Para la planificación de estas tareas se ha intentado seguir una metodología ágil. Dado que el proyecto dispone de entregas parciales se ha decidido realizar una épica para cada entrega y trabajar dicha épica en iteraciones de 2-3 semanas con el objetivo de poder generar un incremento para cada entrega parcial. Tras estas entregas se obtendrá la retroalimentación necesaria por parte del tutor del proyecto para poder mejorar en la siguiente iteración y corregir las posibles desviaciones.

Cabe destacar que no se alcanzaron algunas de las épicas propuestas para la evaluación ordinaria, en concreto la Entrega 3. Borrador final y el Predepósito en periodo ordinario. Por este motivo se tuvo que replanificar para tener en cuenta el periodo vacacional y realizar la entrega en el periodo extraordinario.

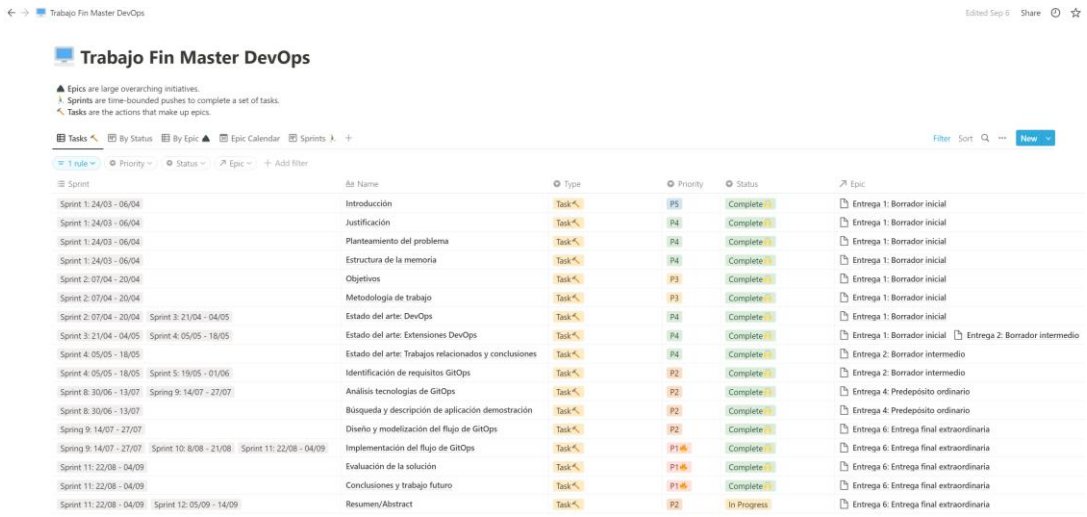
Debido a esto se han creado las siguientes tareas épicas que representan las entregas:

- **Entrega 1.** Borrador inicial.
    - **Sprint 1:** 24/03 – 06/04
    - **Sprint 2:** 07/04 – 20/04
    - **Sprint 3:** 21/04 – 04/05
- **Entrega 2.** Borrador Intermedio
    - **Sprint 4:** 05/05 – 18/05
    - **Sprint 5:** 19/05 – 01/06
- **Entrega 3.** Borrador final.
    - **Sprint 6:** 02/06 – 15/06
    - **Sprint 7:** 16/06 – 29/06

- **Entrega 4.** Predepósito ordinario
    - **Sprint 8:** 30/06 – 13/07
- **Entrega 5.** Depósito ordinario.
    - Sin Sprint.
- **Entrega 6.** Entrega final extraordinaria
    - **Sprint 9:** 14/07 – 27/07
    - **Sprint 10:** 08/08 – 21/08
    - **Sprint 11:** 22/08 – 04/09
- **Entrega 8.** Depósito extraordinario
    - Sin sprint

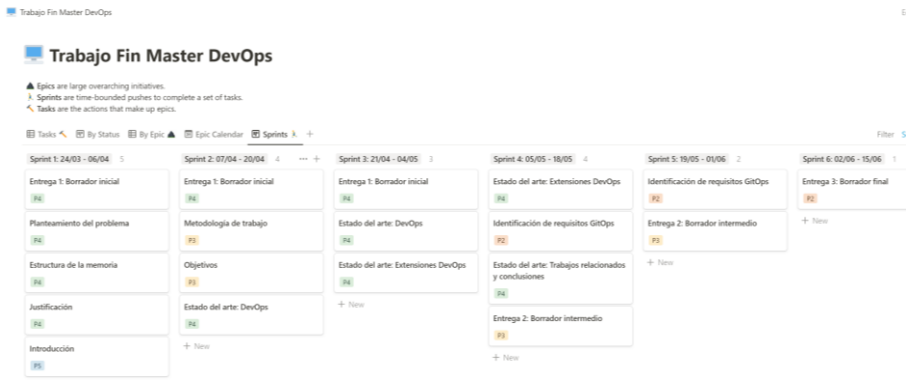
Para organizar el trabajo he utilizado la herramienta gratuita Notion (https://www.notion.so/es-es) y una plantilla predefinida para la gestión de proyectos ágiles. Las siguientes figuras muestran los diferentes paneles de control creados.

Figura 13. *Panel de tareas.*



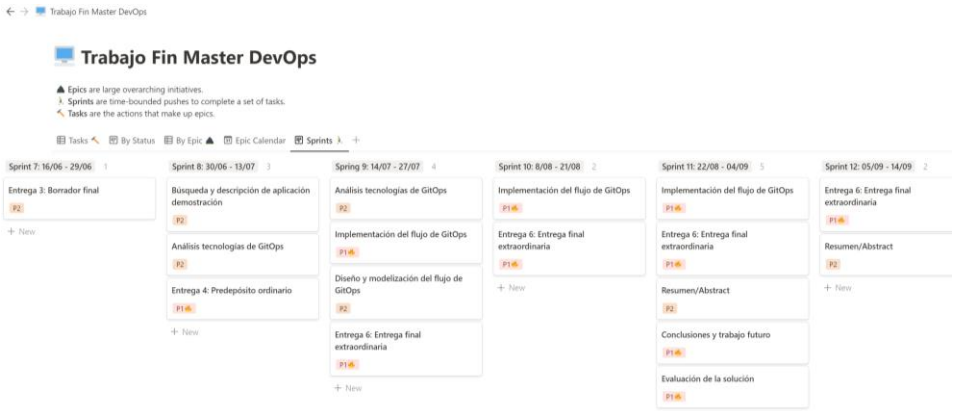
Fuente. Elaboración propia.

Figura 14. *Panel de Sprints 1.*



Fuente. Elaboración propia.

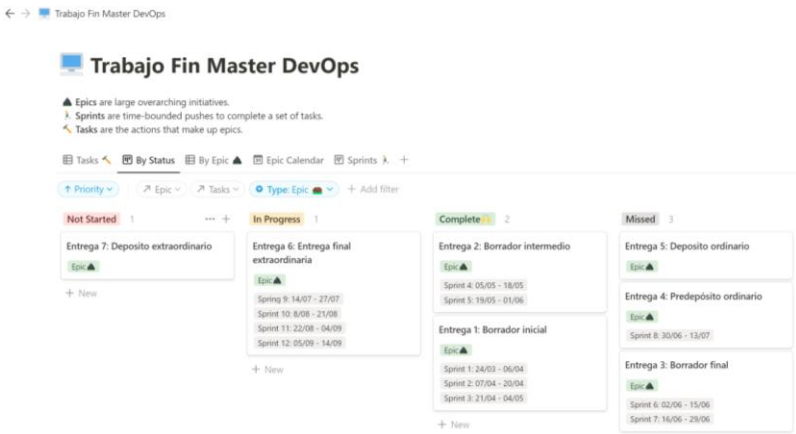
Figura 15. *Panel de Sprints 2.*



Fuente. Elaboración propia.

Como puede apreciarse, las tareas tienen una prioridad asignada. Además, las épicas también tienen un estado. Se ha tenido en cuenta las épicas fallidas, etiquetas como *missed.*

Figura 16. *Panel de estado de épicas.*



Fuente. Elaboración propia.

4. Desarrollo específico de la contribución

En este capítulo se identificarán los requisitos necesarios para la implementación de GitOps y se hará un estudio de las tecnologías más utilizadas a la hora de implantarlo.

Adicionalmente, se realizará una descripción de la aplicación de demostración seleccionada.

Por último, se diseñará e implementará un flujo de trabajo de GitOps con algunas de las herramientas más utilizadas hoy en día.

4.1. Identificación de los requisitos para GitOps

La *Cloud Native Computing Foundation* (CNCF) ha creado un grupo de trabajo compuesto por grandes empresas del sector como Weaveworks, Amazon, GitHub, etc. para establecer unas bases, formalizar y unificar los conceptos sobre GitOps (Garfield & Rigby, 2021). Este grupo ha establecido unos requisitos mínimos que deben cumplirse a la hora de operar con sistemas para que esta operativa pueda considerarse una implementación de GitOps. Estos requisitos son los siguientes:

- **Declarativo**. Un sistema administrado mediante GitOps debe tener su estado deseado expresado de manera declarativa.
- **Versionado e inmutable**. El almacenamiento del estado deseado debe imponer la inmutabilidad, el control de versiones y mantener un historial completo de las versiones.
- ***Pull* automático**. Los agentes de software recuperan automáticamente las declaraciones de estado de la fuente de verdad.
- **Reconciliación continua**. Los agentes de software monitorizan constantemente el estado real del sistema e intentan aplicar el estado deseado.

Las principales prácticas que son necesarias para implantar este marco de trabajo son las siguientes:

- **Infraestructura como Código (IAC).** Necesaria debido al carácter declarativo que se propone para expresar el estado de la infraestructura.
- **Merge/Pull Request (MR/PR).** Actúan como solicitud de cambio a la hora de actualizar el estado y las configuraciones de la infraestructura. Los miembros del equipo podrán

colaborar aportando comentarios. Además, este mecanismo sirve como registro de auditoria una vez mezclado en la rama principal.

- **Integración y despliegue continuo (CI/CD).** Permite automatizar los cambios que se realizan en el entorno sin que el operador tenga que intervenir. Existen dos modelos que funcionan en modo *push* o *pull*, también son conocidos como sin agente y con agente ya que en el modelo *pull* se utiliza un operador para observar los cambios. El modelo con agente es el recomendado ya que permite detectar y corregir las derivas del estado del clúster.

Debido a las ventajas que proporciona el modelo *pull* respecto al modelo *push* es el que se implementará durante este trabajo ya que es el recomendado para trabajar con clúster de Kubernetes.

Existen cuatro componentes clave que permiten construir un flujo de trabajo de GitOps basado en *pull*.

- **Un repositorio para las aplicaciones**. Contiene el código fuente y los manifiestos de despliegue de estas.
- **Un repositorio para las configuraciones del entorno**. Contiene todos los manifiestos del estado deseado para un entorno. Describe tanto aplicaciones como servicios de infraestructura tales como *brokers* de mensajería, herramientas de supervisión, etc.
- **Un registro de imágenes de contenedores**. Es donde se almacenan las imágenes de las aplicaciones.
- **Un agente u operador ejecutando en un clúster de Kubernetes**. Es el encargado de recuperar las configuraciones y las imágenes para reconciliar el estado del clúster.

Estos componentes son los que serán utilizados durante la realización de este trabajo.

4.2. Análisis de las principales tecnologías de GitOps

Actualmente, las tecnologías asociadas con GitOps están en pleno auge por lo que cada día surgen nuevas alternativas. Sin embargo, existen una serie de herramientas que han alcanzado un grado de madurez suficiente como para considerarse un estándar de facto en la implementación de GitOps. Algunas de estas herramientas son las siguientes:

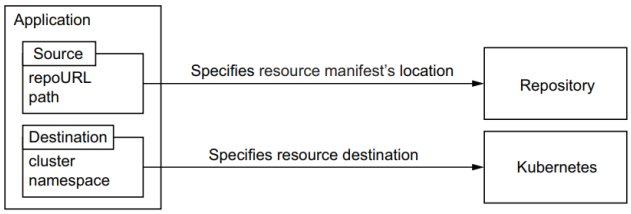
4.2.1. ArgoCD

ArgoCD (ArgoCD, 2022a) es una herramienta de código abierto que proporciona entrega continua declarativa para Kubernetes. Esta herramienta es parte de un conjunto de herramientas de la misma familia que se centran en la entrega de aplicaciones. Cabe destacar que actualmente está siendo incubado por la Cloud Native Computing Foundation (CNCF).

ArgoCD se basa en dos conceptos clave:

- **Aplicaciones**. Permiten agrupar de manera lógica recursos de Kubernetes y definen el origen del manifiesto (URL del repositorio y directorio donde se encuentran los ficheros de manifiesto) y el destino de los recursos (Clúster y Namespace de Kubernetes donde se desplegarán los recursos), *ver Figura 17*. Además, también dispone de otras propiedades que representan el estado de sincronización del recurso con el repositorio y su salud.

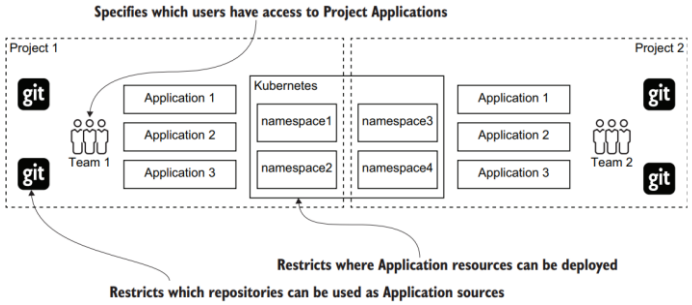
Figura 17. *Estructura de una aplicación de ArgoCD.*



Fuente. Yuen et al. GitOps and Kubernetes. Capítulo 9.

- **Proyectos**. Proporciona agrupaciones lógicas de aplicaciones permitiendo aislar a los equipos unos de otros y realizar configuraciones de control de acceso con mayor precisión. Gracias a esto se consigue restringir los repositorios de Git o los Clústeres de Kubernetes en los que las aplicaciones pueden ser desplegadas. Es un elemento clave en entornos multitenant.

Figura 18. *Estructura de un proyecto de ArgoCD.*



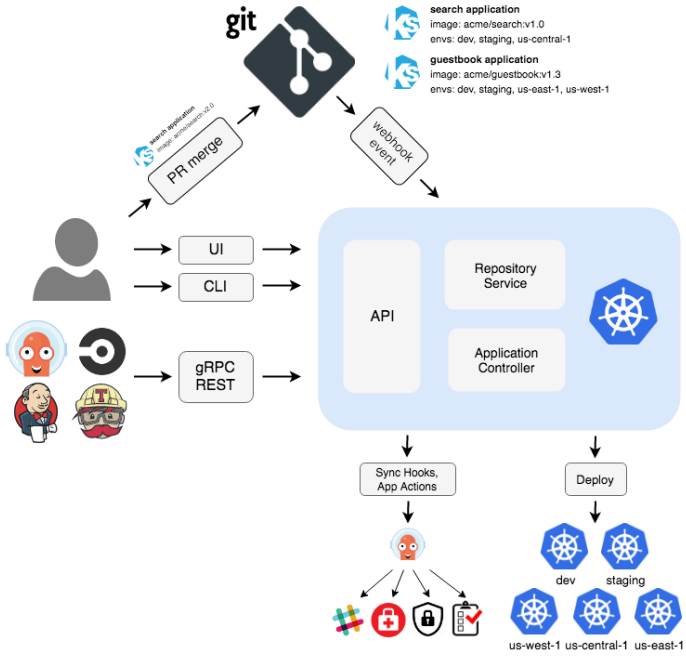
Fuente. Yuen et al. GitOps and Kubernetes. Capítulo 9.

Los componentes principales de ArgoCD son los siguientes:

- ***API server***. Expone el API que consumen la herramienta de interfaz de usuario, la herramienta de consola y servidores de CI/CD. Se encarga principalmente de la gestión de aplicaciones y todo lo que ello conlleva como puede ser la generación de informes de estado, administración de credenciales, etc.
- ***Repository server***. Es el encargado de gestionar la caché local del repositorio que contiene los ficheros manifiestos de las aplicaciones. También genera y devuelve los ficheros de manifiesto cuando dispone de los parámetros de configuración adecuados.
- ***Application controller***. Lleva a cabo la monitorización de las aplicaciones en ejecución y controla que no existan desvíos en la configuración mediante la comparación del estado actual contra el estado almacenado en el repositorio de Git. Además., puede llevar a cabo medidas correctivas que permitan corregir dicho estado.

La Figura 19 ilustra la arquitectura de ArgoCD.

Figura 19. *Arquitectura de ArgoCD.*



Fuente. Documentación oficial de ArgoCD.

Cabe destacar que ArgoCD puede autogestionarse. Para conseguir esto basta que una instancia de ArgoCD vuelva a instalar ArgoCD, de esta manera se consigue corregir derivas en la configuración (ArgoCD, 2021).

4.2.2. Flux 2

Flux versión 2 (Flux, 2022) es una herramienta de código abierto desarrollada por Weaveworks que permite la sincronización entre los clústeres de Kubernetes y los repositorios de configuración y que además automatiza las actualizaciones de los recursos cuando hay que desplegar nuevas versiones. Al igual que ArgoCD, Flux 2 también es un proyecto incubado por la Cloud Native Computing Foundation (CNCF).

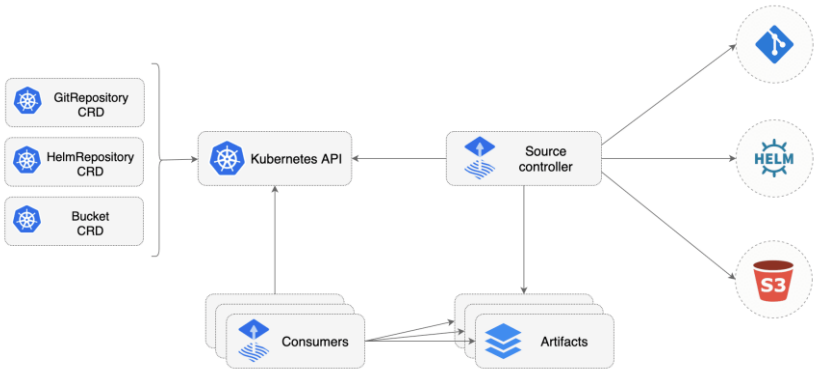
Flux 2 es el heredero de la herramienta original, Flux, que fue desarrollada por la misma empresa. Esta nueva versión se ha construido desde cero para integrarse a la perfección con el API de Kubernetes mediante su extensión a través de la definición de recursos personalizados (*Custom Resource Definition* – CRD) y para proporcionar la capacidad de gestionar múltiples clústeres de Kubernetes. Esta característica, que en la versión anterior no estaba soportada, es fundamental para conseguir una mejor gestión de los entornos empresariales y por la que competidores como ArgoCD han conseguido una gran cuota de mercado.

Cabe destacar que la versión 1 de Flux se sigue usando, sin embargo, únicamente recibe actualizaciones de seguridad y bugs. Eventualmente dicha versión será completamente deprecada por lo que se recomienda la migración a versión 2 siempre que sea posible.

Los componentes clave de Flux 2 son los siguientes:

- ***Source controller***. Permiten crear las fuentes de datos mediante los *Custom Resource Definitions* (CRD). Dichas fuentes definen el repositorio o almacenamiento de origen que contiene el estado deseado y los requisitos de acceso al mismo (protocolo, credenciales, selector de versión, etc.). Flux consume este estado y genera artefactos con los que trabaja sobre el clúster. Estos orígenes se escanean periódicamente en busca de cambios que coincidan con los criterios establecidos.

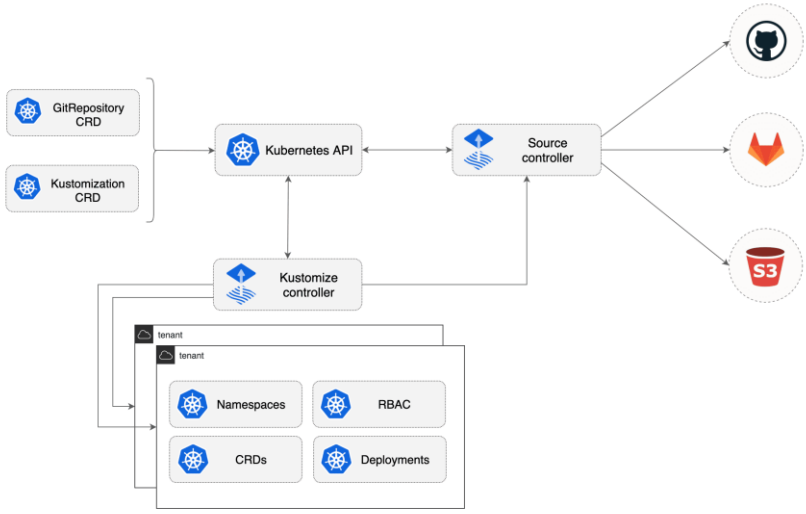
Figura 20. *Arquitectura del source controller.*



Fuente. Documentación oficial de Flux.

- ***Kustomize controller*.** Es el encargado de ejecutar el emplantillado de los ficheros de manifiesto de Kubernetes utilizando la herramienta *Kustomize* (https://kustomize.io/) para poder ajustarlos a las necesidades que se requieran. Un ejemplo es el de sobrescribir la versión de la imagen de un fichero.

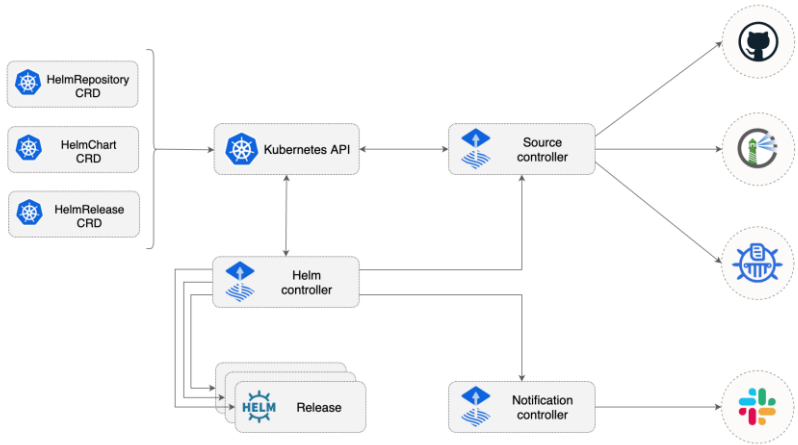
Figura 21. *Arquitectura del kustomize controller.*



Fuente. Documentación oficial de Flux.

- ***Helm controller.*** Permite utilizar los *charts* de la herramienta Helm (https://helm.sh/) como fuente. Las *releases* de los *charts* se instalarán en el clúster de Kubernetes lo que permitirá su gestión mediante Helm por lo que se podrán actualizar o desinstalar cuando sea necesario.

Figura 22. *Arquitectura del helm controller.*

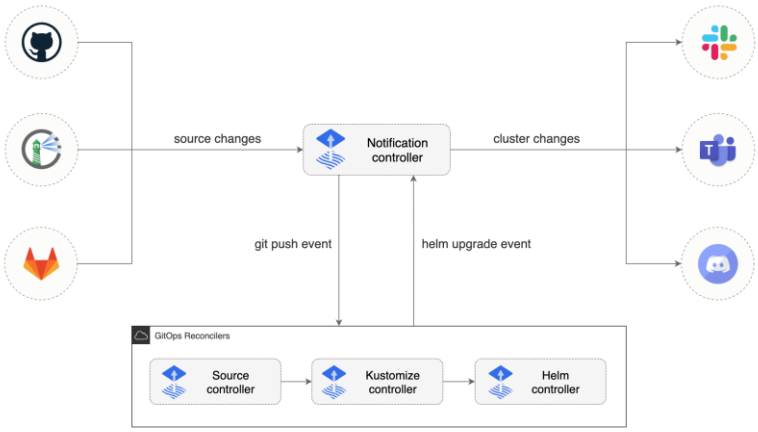


Fuente. Documentación oficial de Flux.

- ***Notification controller.*** Permite gestionar los eventos del sistema. Los eventos entrada provenientes de sistemas externos (repositorios remotos de Git, servidores de CI/CD, etc.) permiten notificar a los controladores sobre cambios en el código fuente.

Además, también permite manejar los eventos de salida que generan los controladores para enviarlos hacia otros sistemas (Slack, email, etc.).

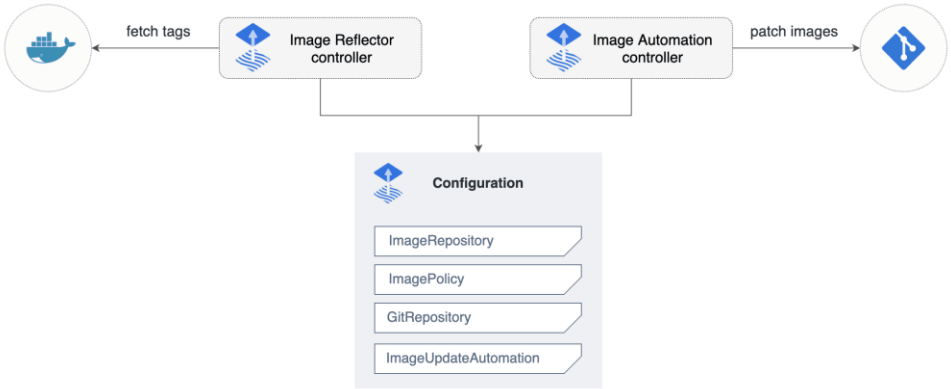
Figura 23. *Arquitectura del notification controller.*



Fuente. Documentación oficial de Flux.

- ***Image automation controller.*** Se encarga de gestionar el escaneo de las imágenes del repositorio de imágenes y gestionar la actualización de los ficheros de manifiesto de Kubernetes y su envío al repositorio remoto de Git.

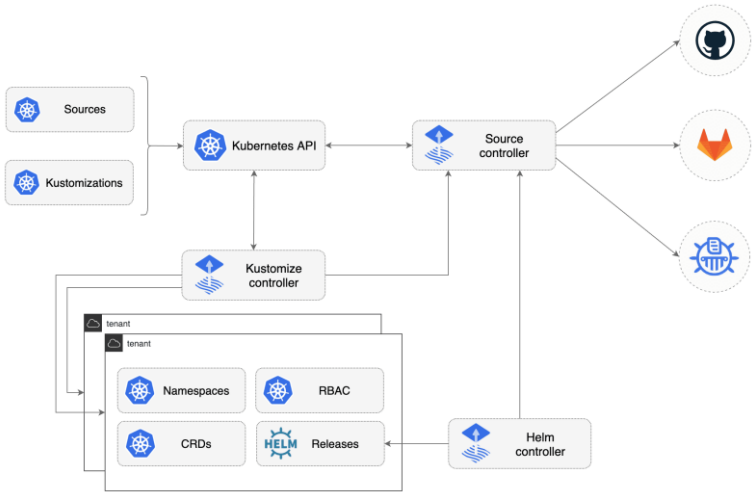
Figura 24. *Arquitectura del image automation controller.*



Fuente. Documentación oficial de Flux.

La Figura 25 muestra la arquitectura global de Flux versión 2.

Figura 25. *Arquitectura global de Flux 2.*



Fuente. Documentación oficial de Flux.

4.2.3. JenkinsX

JenkinsX (JenkinsX, 2022) es una herramienta de código abierto que busca simplificar los procesos complejos en conceptos que puedan entenderse y adoptarse fácil y rápidamente. Su principal objetivo es del automatizar y acelerar la integración y entrega continua para que los desarrolladores puedan centrarse en la creación del software.

Cabe destacar que Jenkins X aboga por la simplicidad por lo que para usarlo no se requieren grandes conocimientos sobre Kubernetes. Además, esta herramienta se apoya sobre una gran cantidad de herramientas bien conocidas para proporcionar las funcionalidades requeridas, es decir, actúa como un pegamento que integra diferentes herramientas para conseguir construir un proceso de integración y entrega continua eficiente y que se amolde a las necesidades que tenemos.

De las herramientas en las que se basa Jenkins X cabe destacar Tekton (Tekton, 2022) ya que es la base que utiliza para construir sus *pipelines*. Esta herramienta proporciona una serie de *Custom Resource Definition (CRDs)* para Kubernetes que permiten el crear un sistema de integración y entrega continua dentro de un clúster. Las siguientes entidades son las principales que define Tekton y las que utilizará Jenkins X.

- **Task.** Define un conjunto de pasos que ejecutan herramientas de construcción o entrega que reciben entradas y generan salidas.

- **TaskRun.** Instancia una tarea para ejecutarla con unas entradas, salidas y parámetros específicos.
- **Pipeline.** Define un conjunto de tareas que llevan a cabo un objetivo de compilación o entrega.
- **PipelineRun.** Instancia un *pipeline* para que ejecute con unas entradas, salidas y parámetros concretos.

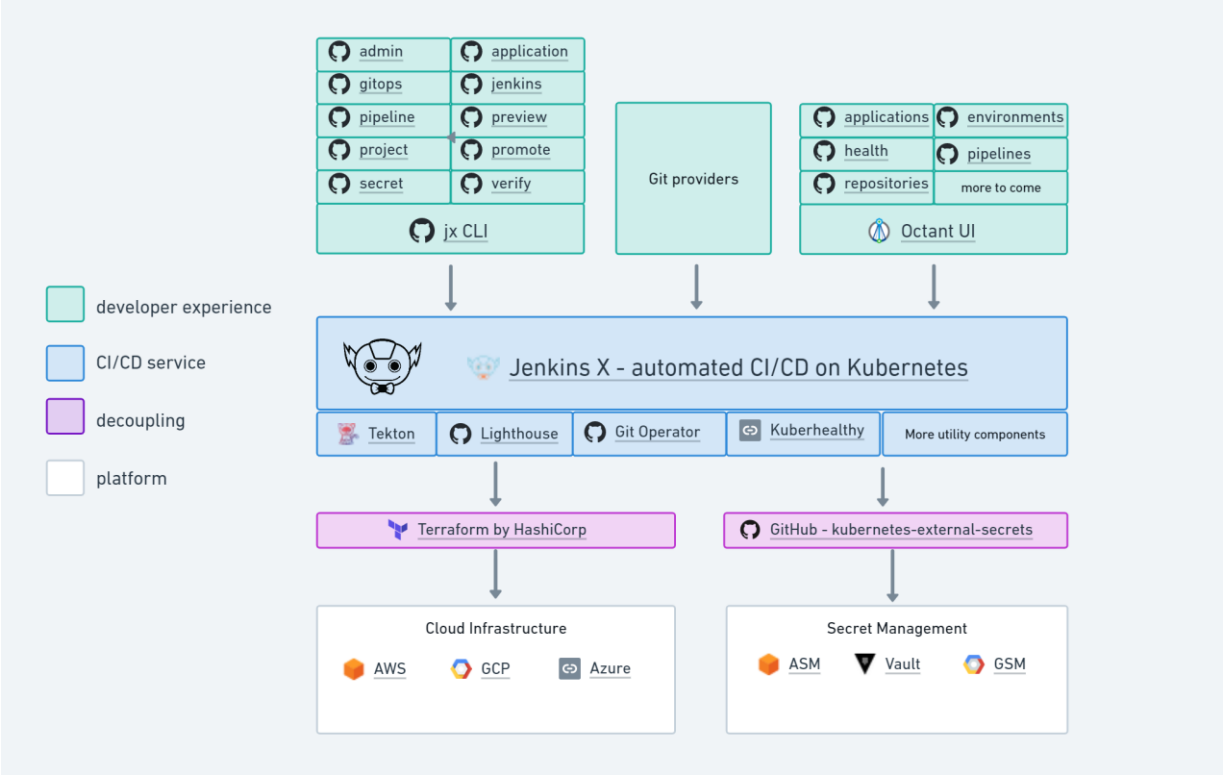
Además, los principales componentes de Jenkins X son los siguientes:

- **Source repositories.** Son repositorios de gestión de código que han sido configurados para ejecutar un pipeline de CI/CD. Incluyen información relativa al proveedor del repositorio, organización, nombre y URL.
- **Environments.** Representa el entorno en el que vive la aplicación (desarrollo, pruebas, producción, etc.). Se encuentran acotados a un *namespace* de Kubernetes y extienden el mismo con metainformación. Los entornos pueden encontrarse en el mismo clúster o en clústeres remotos. Es donde ejecutarán los pipelines. Cabe destacar que existen varios tipos y que las aplicaciones pueden promocionarse entre entornos de manera automática o manual.
- **Pipeline activities.** Se componen de pasos que pueden ser de distintos tipos y que permiten alcanzar objetivo de la actividad. Estos pasos son:
    - **Stage.** Se asocia con la parte de integración continua de Jenkins X. Permite representar el flujo de descarga del código, compilación, etc.
    - **Preview.** Se encarga de crear entornos de *preview* como parte de las *Pull Request* que se realicen.
    - **Promote.** Es la responsable de desplegar una versión de una aplicación en un entorno concreto.

Jenkins X utiliza los componentes mencionados anteriormente para desplegar las aplicaciones en los clústeres de los distintos entornos. Para llevar a cabo esta tarea utiliza una serie de ficheros que permiten definir las reglas de disparo de los *pipelines* definidos en las *Pull request*, los pasos a seguir cuando se abre una *Pull request* contra la rama principal del repositorio y las diferentes tareas que se deben ejecutar cuando se lleva a cabo un cambio en la rama principal del repositorio.

La Figura 26 muestra una perspectiva general de la arquitectura de Jenkins X.

Figura 26. *Arquitectura de JenkinsX.*



Fuente. Documentación oficial de JenkinsX.

4.3. Descripción de la aplicación de demostración

Para llevar a cabo este proyecto es necesario disponer de una aplicación que permita realizar las pruebas una vez que el flujo haya sido implementado. El desarrollo de esta aplicación queda fuera del ámbito de este trabajo. Debido a esto se ha buscado una aplicación que cumpla una serie de requisitos básicos que suelen cumplir las aplicaciones que se desarrollan ser desplegadas en Kubernetes.

The Twelve-Factor App (Wiggins, 2017) es una metodología que facilita la construcción de aplicaciones SaaS (Software as a Service) que define una serie de buenas prácticas y requisitos para el desarrollo, construcción y puesta en producción de dichas aplicaciones. Los principales objetivos de las aplicaciones *Twelve-factor* son los siguientes:

- Se usa el formato declarativo en las automatizaciones de la configuración con el objetivo de que estén auto documentadas.

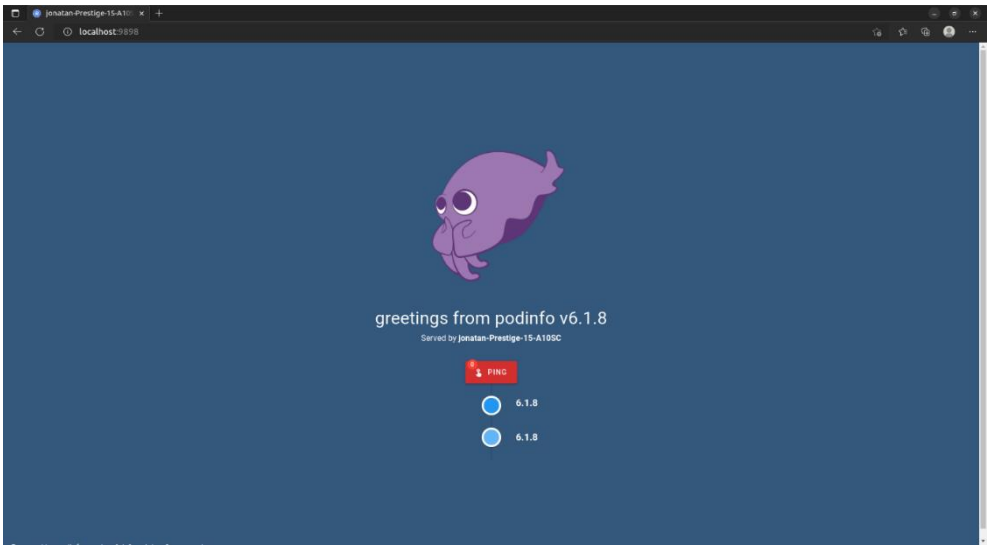
- Cumplen un contrato con el sistema operativo sobre el que ejecutan facilitando la portabilidad.
- Son fácilmente desplegables en las plataformas basadas en la nube.
- Eliminan las diferencias entre los distintos entornos de la compañía facilitando el despliegue continuo y mejorando la robustez de los procesos.
- Son fácilmente escalables sin que se necesiten cambios bruscos en la arquitectura o las herramientas.

La aplicación escogida se llama *podinfo*. Es una aplicación web desarrollada en Go y basada en microservicios que permite ilustrar las mejoras prácticas al trabajar con estos. Cabe destacar que es una aplicación muy usada para llevar a cabo demostraciones en diferentes proyectos de la *Cloud Native and Computing Foundation*. Su código fuente puede encontrarse en GitHub (https://github.com/stefanprodan/podinfo).

Esta aplicación se distribuye como una imagen de *Docker* y para facilitar su despliegue en Kubernetes también se dispone de un *Chart* de *Helm* y de unas plantillas de *Kustomize*. Cabe destacar que la aplicación puede desplegarse como si fuese una aplicación monolítica o una aplicación multicapa estructurada en 2 capas, una para el *frontend* y otra para el *backend*. Gracias a esto se dispone de la capacidad para escalar cada capa de manera independiente lo que permitirá obtener un ahorro de costes ya que se puede realizar escalado de la capa que sea necesaria.

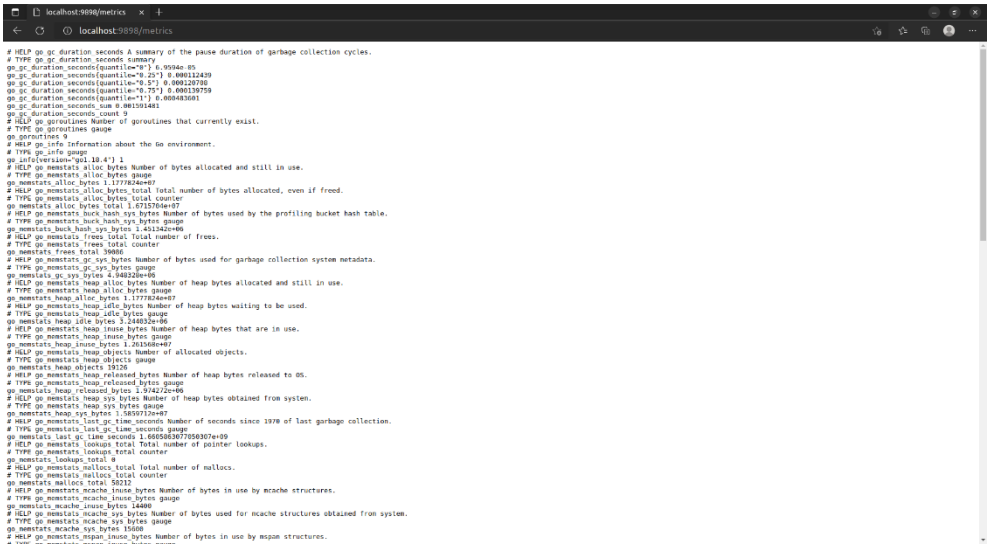
Entre las funcionalidades de *Podinfo* destacan su sencilla interfaz web, ver Figura 27 y su API REST que permite exponer una gran cantidad de metainformación entre la que destaca información sobre métricas, salud y disponibilidad ver Figura 28.

Figura 27. *Página principal de Podinfo.*



Fuente. Podinfo. Elaboración propia.

Figura 28. *Métricas de Podinfo.*



Fuente. Podinfo. Elaboración propia.

Por último, cabe destacar que todas estas funcionalidades se han probado utilizando la última imagen de Docker en un entorno local con el fin de evaluar si la aplicación cumplía los requisitos necesarios.

4.4. Diseño e implementación del flujo de trabajo de GitOps

Tal y como se explica en la sección 2.1.1, el ciclo de vida de DevOps consta de una serie de etapas que facilitan el desarrollo, puesta en producción y operación de las aplicaciones desarrolladas. A fin de conseguir estos beneficios se llevan a cabo una serie prácticas,

explicadas en la sección 2.1.4, que promueven la automatización de las tareas, la colaboración entre los integrantes del equipo y la mejora continua.

Dentro de dichas prácticas destacan la Integración, Entrega y Despliegue Continuo (CI/CD). GitOps es un marco operacional que se centra en el Despliegue Continuo (CD – Continuous Deployment) de las aplicaciones y de la infraestructura. Cabe destacar que cuando se habla de infraestructura no se hace referencia a máquinas virtuales o servidores ya que Kubernetes actúa como abstracción de dichos componentes. El termino infraestructura hace referencia a cualquier dependencia que la aplicación pueda tener y que deba ser desplegada en el clúster tales como servidores web o bases de datos.

4.4.1. Diseño del flujo

Para poner la aplicación en producción de manera fiable es necesario que se puedan llevar a cabo las prácticas mencionadas anteriormente. Para conseguir esto se ha realizado el diseño de un flujo de CI/CD que permita ilustrar los diferentes pasos que tendrán que ejecutarse para construir la aplicación y que posteriormente esta sea desplegada en el clúster de Kubernetes.

4.4.1.1. Estrategia de ramificación

Cabe destacar que para implementar un flujo de GitOps se dispone de dos repositorios de código distintos a los cuales se les pueden aplicar diferentes estrategias de ramificación.

Repositorio de código fuente

La estrategia de ramificación que se propone para el repositorio que contiene el código fuente de la aplicación es "OneFlow" (Ruka, 2017) con las opciones de no aplicar el *"fast-forward"* y realizando *"rebase"*. Esta es una estrategia sencilla que facilita una alta colaboración entre los desarrolladores cuando los equipos son pequeños y medianos.

La estrategia consiste en generar una rama para cada nueva característica o solución de bugs que acabará generando una *Pull Request*. Si dicha solicitud se acepta, el código se mezclará en la rama principal denominada *máster*.

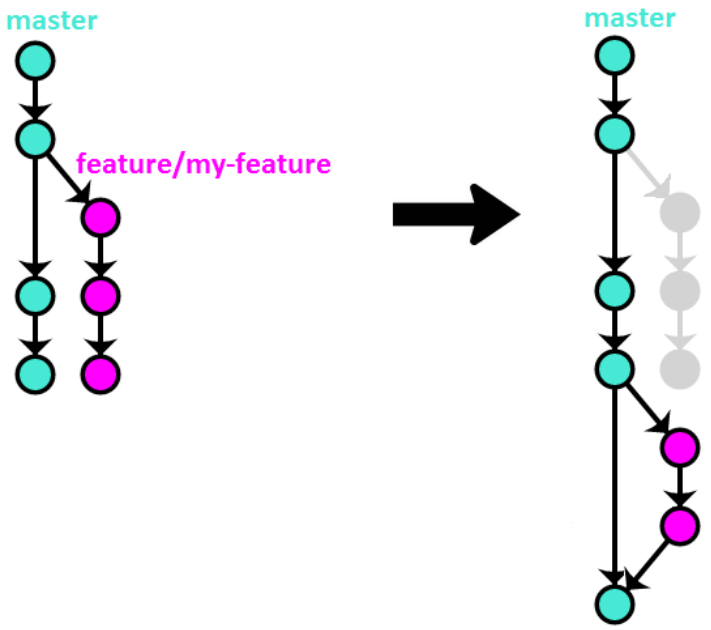
La opción de *"rebase"* permite que las ramas dispongan siempre de la última versión existente en la rama principal por lo que las pruebas de integración se ejecutarán siempre con la última versión disponible el código ver Figura 29 y además se podrán detectar y solucionar los conflictos antes de fusionar el código.

Respecto a la opción de deshabilitar el avance rápido, *"no fast forward"*, permite que se genere un nuevo *commit* en la rama principal. Esto facilita las regresiones en caso de ser necesario ya que basta con volver al *commit* de fusión anterior para deshacer los nuevos cambios.

Gracias a este método de trabajo se consigue que la rama principal, *máster*, nunca contenga código que no compile o no funcione ya que mediante las *Pull Request* se habrán detectado posibles conflictos de fusión, pruebas con resultados erróneos y, además, algún compañero del equipo habrá tenido que validar el código antes de que la solicitud sea aprobada.

Una vez que la solicitud de cambio es fusionada con la rama principal se genera una nueva versión de la aplicación que puede desplegarse en el entorno de desarrollo por lo que se incrementará la versión de la aplicación en el repositorio de configuración.

Figura 29. *Feature branching usando OneFlow.*



Fuente. Ruka, Adam. OneFlow – a Git branching model and workflow. End of Line blog.

Repositorio de configuración

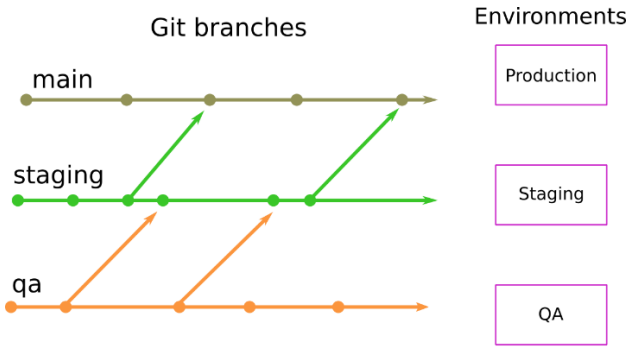
Respecto al repositorio de manifiestos o configuración utilizado para la implementación de GitOps existen múltiples maneras de organizar el código cada una con sus ventajas y desventajas.

Cabe destacar que el objetivo de este repositorio es el de describir las configuraciones y diseños en ficheros de texto por lo que si no se gestiona correctamente puede ser que se

acabe lidiando con una gran cantidad de ficheros para los distintos entornos creando un problema que antes no se tenía.

Es natural pensar que una estrategia en la que existan tantas ramas como entornos facilitará la gestión de las configuraciones, ver Figura 30, especialmente las promociones entre entornos gracias al uso de las *Pull Request*.

Figura 30. *Ramificación basada en entornos.*



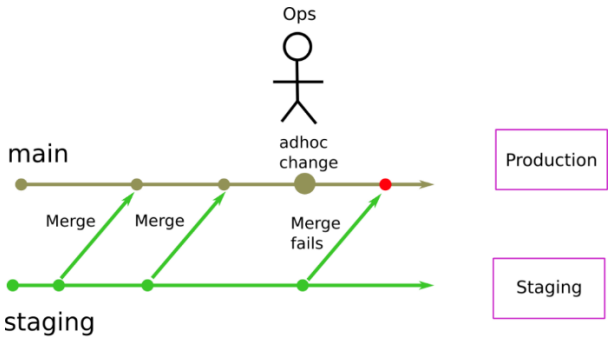
Fuente. Kapelonis, Kostis. Stop Using Branches for Deploying to Different GitOps Environments. Codefresh.

Sin embargo, esta estrategia se considera un anti-patrón que acaba generando una gran carga de trabajo cuando se dispone de un gran número de entornos (Kapelonis, 2021):

- **No siempre basta con una simple fusión para promocionar el código entre entornos.** Utilizar este mecanismo no permite llevar un control granular de los cambios lo que puede provocar que se añadan cambios no deseados al entorno o estos estén desordenados. Un ejemplo de esto es el número de réplicas de una aplicación en cada entorno. No se quiere cambiar el número de réplicas del entorno de producción por el de QA por lo que se necesita aplicar un control mucho más granular que un simple "*git merge*".
- **Se pueden generar derivas en la configuración fácilmente.** Si se llevan a cabo cambios en el entorno de producción de manera local y estos no son trasladados a los otros entornos, entonces se podrían llegar a producir conflictos a la hora de llevar a cabo la promoción de las aplicaciones. Ver Figura 31.
- **El modelo de una rama para cada entorno va en contra de Kustomize y Helm.** Estas dos herramientas son muy utilizadas en los entornos de Kubernetes. Aunque son herramientas distintas, en líneas generales funcionan de manera similar. Ambas

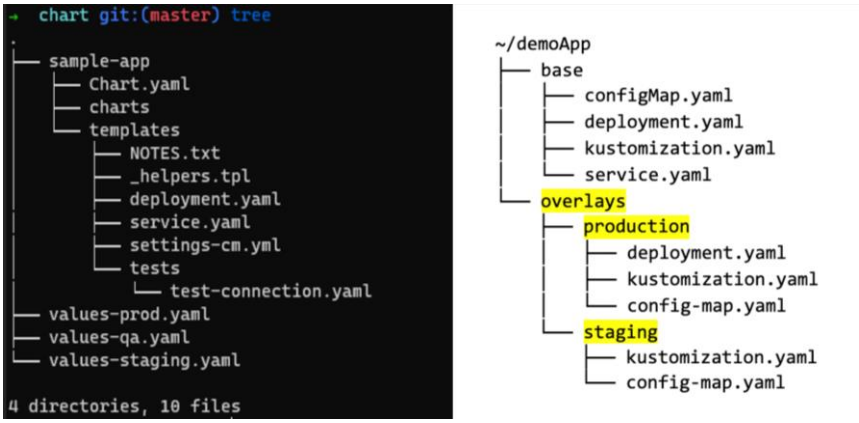
permiten definir una base parametrizable a la que se le puede aplicar distintos cambios para obtener como resultado el artefacto que se aplicará al clúster. Estas herramientas almacenan los valores de los distintos entornos en ficheros que se organizan en directorios o mediante su nombre. Ver Figura 32.

Figura 31. *Cambios locales usando ramas basadas en entornos.*



Fuente. Kapelonis, Kostis. Stop Using Branches for Deploying to Different GitOps Environments. Codefresh.

Figura 32. *Estructura de los entornos. Helm vs Kustomize.*



Fuente. Kapelonis, Kostis. Stop Using Branches for Deploying to Different GitOps Environments. Codefresh.

Por estos motivos, la estrategia de ramificación del repositorio de configuración puede ser la misma que la del repositorio de código, “OneFlow”. Se utilizará una rama principal que contendrá los ficheros base y de los entornos y ramas efímeras que se utilicen para modificar la estructura y poder revisar los cambios entre los entornos antes de que se apliquen.

Respecto a la organización del repositorio es recomendable seguir las prácticas propuestas por herramientas como Helm. Las recomendaciones son las siguientes:

- Estructura del *Chart* base en una carpeta *templates* que contiene las plantillas base.
- Distintos ficheros de valores para los entornos.

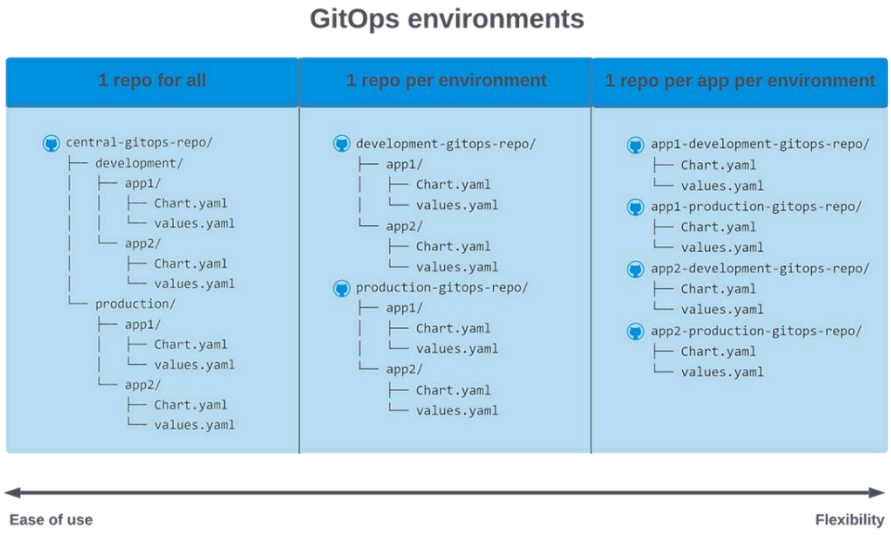
- **Desarrollo**: values.yaml.
- **QA**: values-qa.yaml
- **Producción**: values-producción.yaml

Con esta estructura, los manifiestos de un entorno se generarán mediante la mezcla de las plantillas base con los valores y valores de dicho entorno.

Además, la promoción entre los entornos consistirá en trasladar los valores de los ficheros entre los distintos ficheros existentes. Gracias al uso de *"feature branches"* se podrán generar *Pull Request* que podrán ser validadas y fusionadas con la rama principal que representará el estado deseado de los distintos entornos.

Cabe destacar que este concepto puede llevarse a cabo utilizando múltiples repositorios para las aplicaciones y los entornos. Sin embargo, se ha preferido optar por la simplicidad y facilidad de uso.

Figura 33. *Diferentes estructuras para repositorios de configuración en GitOps.*



Fuente. Fokker, Hijmen. How to manage GitOps environments at a scale (a technical guide). Pionative.

4.4.1.2. Modelización del flujo

Aunque GitOps esté centrado en el Despliegue Continuo he considerado necesario incluir el flujo de la Integración Continua para disponer de una perspectiva holística del proceso.

El flujo de CI/CD es el siguiente:

1. Se parte de un código que está almacenado en el sistema de control de versiones, Git, del desarrollador y que está conectado a un repositorio de código remoto, GitHub en este caso.
2. El desarrollador crea una nueva rama para implementar la característica. Durante la implementación añadirá cambios a dicha rama y los enviará al repositorio remoto. Al terminar la implementación del código y sus pruebas, se abre una *Pull Request* para fusionar la rama con la principal. Esta solicitud permite que los compañeros revisen el cambio y se proporcione retroalimentación al creador.
3. Al abrir la solicitud de fusión, se ejecutará el pipeline de la aplicación que permitirá evaluar que el código esté correcto, las pruebas pasen y se construyan los artefactos. Cabe destacar que únicamente se ha implementado el pipeline asociado a la rama principal. Sin embargo, el pipeline asociado a las ramas y a las *Pull Request* puede llegar a desplegar un entorno efímero similar a desarrollo para ejecutar pruebas más sofisticadas de manera automática. Tras aceptar la fusión, el código se incorporará a la rama principal y la rama de origen podrá ser eliminada.
4. Al llevarse a cabo esta fusión con la rama principal, se ejecutará un pipeline que volverá a comprobar el código, generará la nueva versión de la aplicación, empaquetará la imagen de Docker para enviarla al registro de contenedores y, por último, se actualizará la versión de la aplicación en el fichero de valores del entorno de desarrollo de la aplicación *values.yaml*.
5. En el momento de dicha actualización del fichero en el repositorio de configuración, esta versión se desplegará automáticamente en el clúster de desarrollo por medio de la herramienta de GitOps. Finalmente, la nueva versión estará disponible para su uso.
6. En caso de que querer llevar la aplicación a otro entorno, como producción, se tendrá que crear una rama, modificar la versión del fichero del entorno correspondiente, *values-prod.yaml*, y abrir una *Pull Request*. Tras validar el cambio se podrá aceptar la solicitud de cambios y, una vez en la rama principal, ArgoCD podrá desplegar dicho fichero en el clúster de producción.

4.4.2. Implementación del flujo

Para llevar a cabo la implementación del flujo de trabajo era necesario seleccionar las herramientas y configurar las mismas acorde a las necesidades expuestas en los apartados anteriores.

4.4.2.1. Herramientas escogidas

Los motivos principales que se han tenido en cuenta a la hora de escoger las herramientas utilizadas en este trabajo son los siguientes:

- **Licencia para uso gratuito o versiones de la comunidad.** Era necesario que no fuese necesario pagar para utilizarlas, aunque tuviesen restricciones.
- **Estándares en el sector.** El objetivo es aprender a utilizar herramientas que se están demandando hoy en día a los profesionales del DevOps.
- **Válidas para entornos corporativos.** Para que el trabajo se ejecute en entornos lo más parecidos posible a los empresariales.
- **Que sean de código abierto.** El aporte de la comunidad es fundamental a la hora de documentar, resolver dudas y evolucionar la herramienta.

Las herramientas que han sido escogidas para el desarrollo de este trabajo son las siguientes:

- **Terraform.** GitOps pretende desplegar aplicaciones e infraestructura sobre Kubernetes, sin embargo, es necesario llevar a cabo un despliegue inicial del clúster para poder tener un entorno sobre el que trabajar. Además, permite aprovisionar otros recursos de la nube que puedan necesitarse.
- **Azure.** Se ha escogido Azure porque la Universidad proporciona una cuenta de estudiante con crédito para poder trabajar de manera gratuita. Cabe destacar que inicialmente se quiso utilizar Amazon Web Services, pero la cuenta de estudiante no permite automatizar la creación del clúster mediante herramientas como Terraform por lo que se descartó. Dentro de Azure se utilizan los siguientes servicios:
    - **Azure Kubernetes Cluster.** Permite desplegar clústeres de Kubernetes.
    - **Azure Zone DNS.** Es el servicio de nombres de Azure, facilita la gestión de las zonas de DNS.
    - **Azure Key Vault.** Almacén de secretos que proporciona Azure.

- **GitHub.** Es el repositorio de código remoto que se ha seleccionado para almacenar el código fuente del trabajo.
- **GitHub Actions.** Para disponer de un servidor de CI/CD integrado con el repositorio de código remoto. Inicialmente se pensó en desplegar una instancia de Jenkins, pero tras visualizar un seminario impartido en la Universidad sobre GitHub Actions y su fácil integración con GitHub decidí utilizarlo.
- **Docker Hub.** Es el registro de contenedores utilizado para almacenar las imágenes de Docker de la aplicación de demostración presentada en el punto 4.3.
- **Helm.** Es uno de los estándares de paquetización e instalación de aplicaciones sobre Kubernetes ya que permite gestionar su ciclo de vida de manera sencilla. Se utiliza para que ArgoCD instale las distintas aplicaciones que se usan en el clúster.

Con el objetivo de disponer de un clúster lo más cercano a uno de producción se ha decidido instalar, también mediante GitOps y Helm, una serie de herramientas que nos permiten automatizar la gestión de tareas tales como la gestión del dominio de la aplicación, certificados, etc. de las aplicaciones que ejecutan en el clúster. Gracias a esto, se consigue un entorno lo más realista posible.

Las aplicaciones escogidas son las siguientes:

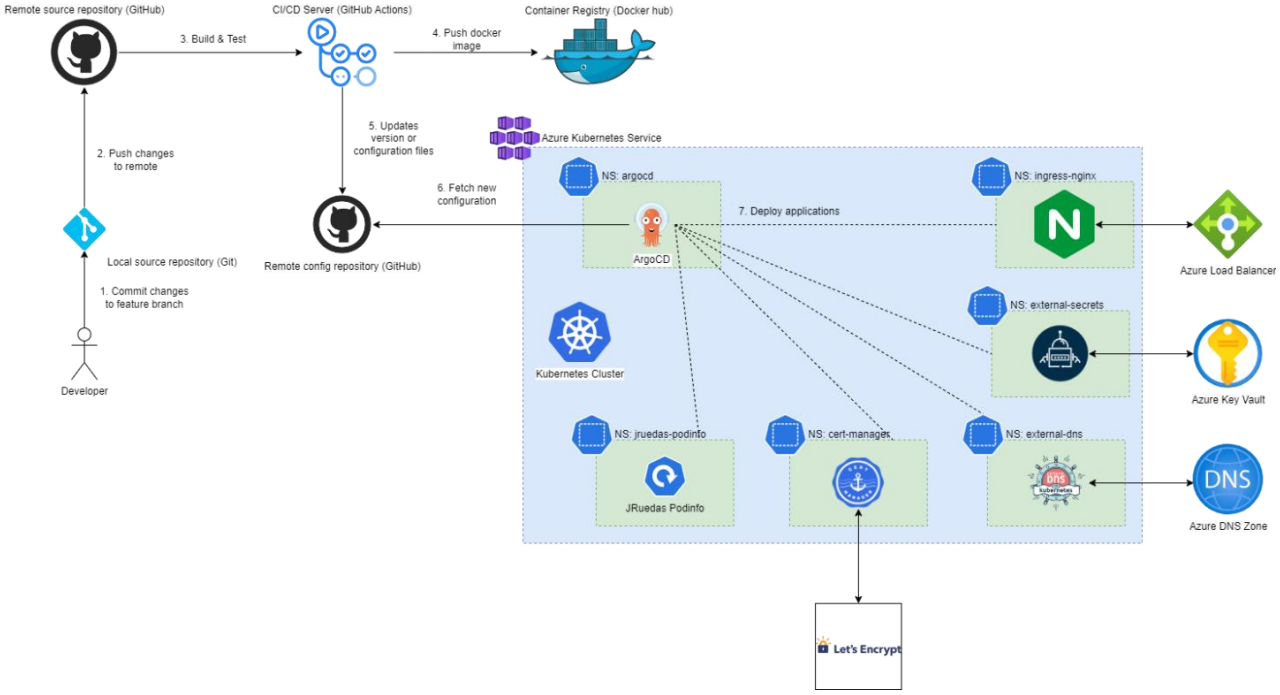
- **Ingress NGINX Controller (Repositorio de GitHub).** Es un *Chart* que despliega un servidor *NGINX* que trabaja como un *Ingress Controller* para el clúster. Gracias a él se pueden exponer las aplicaciones al público utilizando únicamente un balanceador de carga y realizar proxy inverso. Además, al desplegar la aplicación, se encargará de solicitar automáticamente al *Cloud Controller Manager* de Kubernetes que se aprovisione un balanceador de carga.
- **External DNS (Repositorio de GitHub).** Es una aplicación que permite que los recursos de Kubernetes, principalmente los *Servicios* e *Ingress*, puedan ser expuestos a través de un servidor de nombres de dominio. Su principal tarea es la de gestionar los registros DNS del proveedor configurado. En este caso, el proveedor configurado es Azure DNS. Cabe destacar que Azure DNS no proporciona un nombre de dominio, este debe adquirirse por separado y configurarse para que utilice los servidores nombres de Azure. Para este trabajo he utilizado mi nombre de dominio personal (https://jruedas.dev/).

- **Cert Manager (Repositorio de GitHub).** Se encarga de la gestión de los certificados en Kubernetes. Gracias a esta aplicación se puede simplificar y automatizar el proceso de la securización de las aplicaciones mediante HTTPS ya que puede configurarse para utilizar el protocolo de gestión de certificados ACME. En este caso, se ha utilizado la página de Let's Encrypt (https://letsencrypt.org/es/) para obtener certificados válidos para producción de manera gratuita. Gracias a esto, las aplicaciones serán accesibles mediante HTTPS utilizando un navegador web.
- **ArgoCD (Repositorio de GitHub).** ArgoCD es una de las muchas herramientas que han surgido para implementar GitOps. En el apartado 4.2.1 se puede ver una descripción de esta herramienta. Cabe destacar que ArgoCD se instala dos veces. Tras aprovisionar el clúster es necesario que Terraform también instale ArgoCD para iniciar el proceso de GitOps. Tras empezar el flujo, ArgoCD se vuelve a instalar a sí mismo. Esto permite que la aplicación se autogestione de manera que se corrijan las derivas de su propia configuración.
- **External Secrets (Repositorio de GitHub).** Esta aplicación permite gestionar los secretos que están almacenados en bóvedas de manera segura. En este caso, se ha escogido el servicio Azure Key Vault como almacén ya que se integra perfectamente con el servicio del clúster de Kubernetes. La gestión de las credenciales a la hora de implementar GitOps es un punto clave ya que se pretende que toda la configuración se encuentre bajo revisión en los repositorios de código. Este método de trabajo causa un problema muy grave de seguridad ya que se podrían exponer credenciales. Debido a esto, el uso de aplicaciones con estas características es obligatorio. Además, se consigue una centralización de las credenciales que utilizan las aplicaciones.

4.4.2.2. Arquitectura de la solución

A continuación, puede verse un diagrama de arquitectura de la solución planteada.

Figura 34. *Diagrama de arquitectura del flujo.*



Fuente. Elaboración propia.

Tal y como puede verse en la Figura 34, el flujo es el siguiente:

1. El desarrollador lleva a cabo el desarrollo de una característica que va creando en su repositorio local de Git.
2. Una vez que el trabajo está acabado, lo envía al repositorio remoto.
3. En el repositorio remoto se abre una *Pull Request*. Tras verificar que el código pasa las pruebas, se construye la imagen como versión de prueba y se envía al registro de contenedores de Docker. Por último, el código se fusiona con la rama principal.
4. Tras esta fusión, se vuelve a ejecutar el pipeline y esta vez se genera una versión de release (X.Y.Z). A continuación, se envía la imagen construida al registro de contenedores y se crea una rama (y *Pull Request*) en el repositorio de configuración que actualiza la versión del fichero de valores de desarrollo del *Chart* de Helm.
5. Una vez revisada y aceptada la solicitud de cambios en el repositorio de configuración, ArgoCD detectará que existe un cambio y lo desplegará automáticamente.
6. Por último, cada aplicación desplegada hará solicitudes a distintos recursos de Azure tales como las zonas DNS para aprovisionar recursos en ellos.

4.4.2.3. Configuración del flujo de Integración Continua

Para llevar a cabo el flujo de integración continua lo primero que se ha hecho es crear los repositorios necesarios.

- **Repositorio de código fuente.** Se ha realizar un *fork* del repositorio de la aplicación de demostración (https://github.com/stefanprodan/podinfo) a mi cuenta personal de GitHub (https://github.com/JRuedas/podinfo) para poder trabajar en el repositorio ya que era necesario adaptar eliminar partes de este que no son relevantes para este trabajo. Es el repositorio en el que trabajarán los desarrolladores. Cabe destacar que, al usar GitHub Actions como servidor de CI/CD, este repositorio contiene los tokens para poder enviar las imágenes de Docker al registro o los cambios al repositorio de configuración. Estos tokens están configurados como secretos del repositorio.
- **Repositorio de configuración.** Es el repositorio que contiene la configuración de los recursos de Kubernetes (https://github.com/JRuedas/tfm-argocd-apps). Se utilizará como fuente de verdad del estado deseado para que ArgoCD lleve a cabo las acciones pertinentes en el clúster y de esta manera se implemente el flujo de GitOps.
- **Repositorio de creación del entorno.** Es un repositorio que se ha creado para contener el código declarativo de Terraform que permite aprovisionar los recursos necesarios en Azure (https://github.com/JRuedas/tfm-terraform-k8s). Cabe destacar que este repositorio permite aprovisionar el entorno e iniciar el flujo de GitOps ya que también instala la primera versión de ArgoCD en el clúster y que posteriormente se instalará a sí misma para autogestionarse.
- **Registro de contenedores.** Es el registro de contenedores que almacena las imágenes de Docker construidas en GitHub Actions para que el clúster de Kubernetes pueda tener acceso a ellas al desplegar los recursos en el mismo (https://hub.docker.com/r/jruedas92/podinfo).

Cabe destacar que todos estos repositorios, ya sean los de GitHub o el de Docker Hub son públicos por lo que se puede replicar el flujo siempre que se disponga de una cuenta de Azure para llevar a cabo el aprovisionamiento.

Servidor de CI/CD

El objetivo principal de DevOps es el de la automatización de los procesos que engloban el ciclo de vida de una aplicación. Para llevar a cabo las prácticas que esta cultura promueve es necesario disponer de herramientas, tales como los servidores de Integración y Entrega Continua, que nos facilitan la construcción, ejecución de pruebas, empaquetado y despliegue de las aplicaciones producción.

El servidor de CI/CD es una de las piezas clave de todas las empresas ya que será el encargado de orquestar la gran mayoría de procesos automáticos relacionados con las aplicaciones y su liberación. Para este proyecto se ha decidido utilizar GitHub Actions ya que GitHub proporciona una capa gratuita que permite evitar tener que lidiar con la gestión del servidor, su instalación y configuración.

Además, no solo se integra perfectamente con GitHub, si no que dispone de un gran número de acciones creadas por la comunidad que pueden usarse para multitud de propósitos. Adicionalmente, es muy fácil configurar de manera segura los tokens de acceso a otros servicios tales como el registro de contenedores. Estos tokens permanecen asociados al repositorio y el propietario es el único que puede visualizarlos. Adicionalmente, estos se inyectan como variables de entorno en el flujo de trabajo por lo que nunca quedan expuestos.

Por último, destaco que ya conocía Jenkins por lo que he considerado que trabajar con GitHub Actions podía ser una buena experiencia ya que hoy en día es una tecnología que está en auge en muchas empresas.

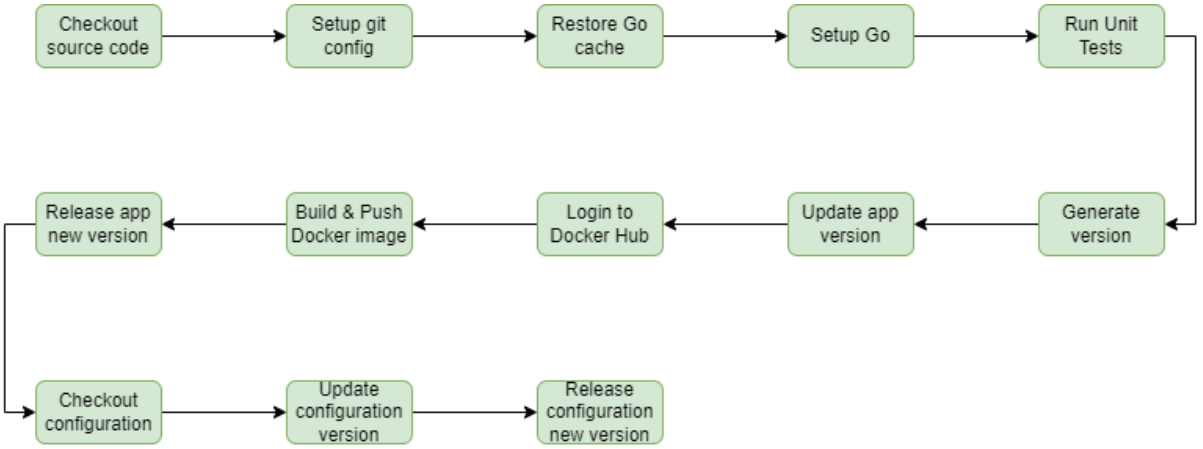
Aunque GitHub Actions es parecido a Jenkins, la nomenclatura no es la misma. En este sistema no existen pipelines, se llaman flujos de trabajo. Dentro de un flujo de trabajo se tienen trabajos que pueden ejecutarse en paralelo. Por último, estos trabajos se componen distintos pasos que ejecutan secuencialmente.

El flujo de trabajo del servidor de CI/CD se configura en el propio repositorio de código fuente donde reside la aplicación. Para realizarlo se debe crear el árbol de carpetas *".github/workflows"* en la raíz del repositorio. En dicha carpeta se almacenen los ficheros, en formato YAML, que definen los flujos.

La aplicación que se ha escogido para demostración incorpora una gran cantidad de características que no se han utilizado o que han tenido que rehacerse, incluido un pipeline de GitHub Actions propio que permite al autor de esta generar las distintas versiones de esta.

Para este trabajo se ha decidido utilizar solamente un flujo de trabajo que, a su vez, dispone de un solo trabajo compuesto por los pasos que pueden verse en la Figura 35.

Figura 35. *Workflow de GitHub Actions.*



Fuente. Ilustración propia.

El anexo B-1 muestra el contenido del fichero que define el flujo de trabajo de GitHub Actions. Además, en el anexo B-2 se puede ver la salida de la ejecución del flujo de trabajo en GitHub.

4.4.2.4. Aprovisionamiento de Kubernetes en Azure

Como se comentó anteriormente, el aprovisionamiento del clúster es una parte muy importante ya que es una condición necesaria para poder poner en marcha el flujo de GitOps.

Por este motivo se ha decidido utilizar Terraform como herramienta para hacer dicho aprovisionamiento. Además, cabe destacar que no solo crea un clúster de Kubernetes, sino que también crea otros servicios que consumirán aplicaciones internas tales como Azure Key Vault o las Zonas de DNS de Azure.

El proyecto se ha organizado por ficheros de manera que cada fichero sea el encargado de crear sus recursos. Los ficheros son los siguientes:

- **Main.tf:** Contiene la lista de proveedores que se utilizarán y las versiones requeridas tanto de estos como de Terraform. Los proveedores utilizados son los siguientes:
    - **Azurerm.** Es el proveedor oficial de Azure. Se utiliza para aprovisionar los recursos en la nube.
    - **Helm.** Es el proveedor oficial de Helm que permite instalar *Charts* en un clúster de Kubernetes. Se utiliza para hacer la instalación inicial de ArgoCD y Cert-manager.

  - **Kubectl.** Es un proveedor creado por el usuario de la comunidad *"Gavinbunney"* y es el proveedor más usado ya que HashiCorp, empresa creadora de Terraform, no proporciona un proveedor para Kubectl. Este proveedor es necesario porque es la dependencia de un módulo que se utiliza para instalar el *Chart* de Cert-manager por primera vez.

- **Provider.tf.** Este fichero contiene las configuraciones de los proveedores que necesitan algún parámetro especial para poder funcionar. La mayoría de las configuraciones son relativas a la ruta donde se almacenará el fichero de configuración de Kubernetes (Kubeconfig) una vez que el clúster se haya aprovisionado. Es importante destacar tanto los proveedores de Helm como de Kubectl no podrán operar hasta que dicho fichero se encuentre en el directorio por lo que harán su fusión una vez que el clúster esté operativo y dicho fichero se haya colocado en la ruta correspondiente.
- **Variables.tf:** Es el fichero que contiene todas las variables del proyecto y sus valores por defecto. Se ha intentado parametrizar la mayor parte de las opciones para poder reutilizar el proyecto a futuro si fuese necesario.
- **Az-resource-group.tf:** Contiene la definición del grupo de recursos de Azure que se utilizará como base y que aglutinará todos los recursos de la nube. El recurso de Terraform utilizado para crear el grupo de recursos es el de *"azurerm_resource_group"*.
- **Aks.tf:** En este fichero es donde está definido el clúster de Kubernetes que se aprovisiona en la nube con las configuraciones correspondientes al número de nodos, su tamaño y la versión de Kubernetes, 1.24. Además, este fichero contiene un segundo recurso que copiará el fichero *Kubeconfig* al directorio *".kube"* de directorio de la cuenta del usuario para que este pueda operar con el clúster y para permitir a los otros proveedores trabajar. Los recursos utilizados en este caso han sido: *"azurerm_kubernetes_cluster"* para el AKS y *"local_file"* para la copia del fichero.
- **Cert-manager.tf:** Este fichero se utiliza para instalar el *Chart* de Helm de Cert-manager y el recurso adicional, *ClusterIssuer*, que permitirá solicitar certificados a Let's Encrypt. El módulo que se usa es de la comunidad y se puede encontrar en el siguiente enlace: https://github.com/terraform-iaac/terraform-kubernetes-cert-manager. El uso de este módulo se debe a que el recurso *ClusterIssuer*, cuando se utiliza un resolutor de

DNS de Azure, solo permite establecer las credenciales de la cuenta como texto en el YAML, no se puede referenciar un secreto del clúster. Esto, en un entorno GitOps no es aceptable por lo que la alternativa es la de instalar esta aplicación y el recurso en tiempo de aprovisionamiento ya que esté modulo permite llevar a cabo esa tarea. Gracias a esto se consigue evitar que las credenciales se almacenen en el repositorio. Posteriormente, ArgoCD volverá a instalar el Cert-manager, pero el recurso *ClusterIssuer*, ya existirá en el clúster por lo que no deberá ser creado. Es necesario mencionar que cuando se usa AWS no existe este problema ya que permite utilizar credenciales almacenadas en un secreto. Por este motivo se ha abierto una *Pull Request* en el repositorio de Cert-manager a fin de que añadan esta característica en un futuro (https://github.com/cert-manager/cert-manager/issues/5412).

- **Dns.tf:** En este fichero es donde se definen los recursos relacionados con el aprovisionamiento de las zonas DNS de Azure. Con el recurso *azurerm_dns_zone* se crea una nueva zona de DNS donde se almacenarán los registros y mediante el recurso *azurerm_role_assignment* se proporcionan permisos al clúster para que pueda contribuir a la zona. Es decir, para que pueda crear, borrar o actualizar los registros DNS. Por último, se genera un fichero local (recurso *local_file*) que contiene los servidores de nombres asignados. Esto es importante ya que como se comentó, Azure no proporciona la capacidad de crear dominios, solo de manejarlos. Por lo tanto, esos servidores de nombres deberán configurarse en el proveedor del dominio. En mi caso, Google domains (https://domains.google/intl/es_es/). Este sitio proporciona una consola web que permite llevar a cabo la operación del cambio de servidores, aunque por desgracia todavía no se puede hacer de manera automática.
- **Vault.tf:** Contiene los recursos necesarios para aprovisionar y configurar Azure Key Vault, el gestor de secretos de Azure. Uno de los retos de GitOps es la gestión de secretos ya que estos no pueden almacenarse en los repositorios de Git. Hoy en día están surgiendo aplicaciones que permite gestionar esto de diferentes maneras. En este caso, la herramienta que instalará ArgoCD podrá conectarse a la bóveda que se aprovisiona y generar los secretos correspondientes. Para aprovisionar esta bóveda se ha utilizado el recurso *azurerm_key_vault_access_policy*. Además, también se tendrán que crear unas políticas para que el usuario de Terraform pueda crear y eliminar secretos mientras que el usuario del clúster únicamente pueda recuperarlos. Para

crear estas políticas se han utilizado los recursos *azurerm_key_vault_access_policy*. Por último, se aprovisiona un secreto, mediante el recurso *azurerm_key_vault_secret*, que contendrá información para que la aplicación External DNS pueda actualizar las zonas DNS cuando se instalen aplicaciones que necesiten un dominio.

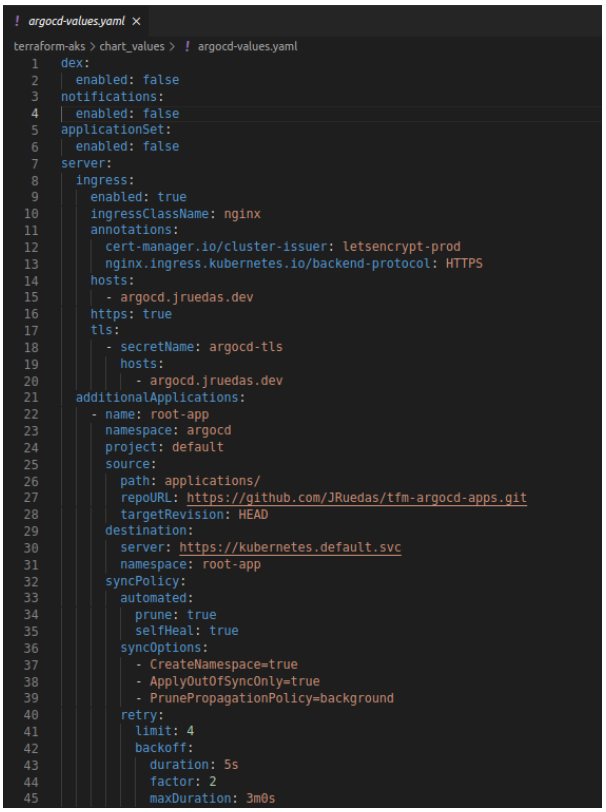
- **Argocd.tf:** Este fichero contiene el recurso que llevará a cabo la instalación inicial de ArgoCD. Dicha instalación se realiza con Helm y se parametriza con una serie de valores que están definidos en una carpeta llamada *chart_values* y que permiten realizar la configuración inicial de ArgoCD y de la aplicación padre que instalará el resto de las aplicaciones.

Respecto a los valores de configuración de ArgoCD se encuentran en el fichero *argocd-values.yaml* que se encuentra dentro de la carpeta *chart_values* del repositorio. Este fichero es leído por el proveedor de Helm de Terraform a la hora de aplicar el *Chart* de Helm. Los valores de configuración se muestran en la Figura 36.

Tal y como puede verse, a la hora de instalarse se define una aplicación adicional que se desplegará tras instalar la herramienta. Esta es la aplicación raíz que permite desplegar el resto de las aplicaciones.

Estos valores son los mismos que se utilizarán cuando ArgoCD se auto instale automáticamente. Se explicarán en más detalle en el apartado 4.4.2.5

Figura 36. *Configuración de ArgoCD.*



Fuente. Elaboración propia.

Lo primero que se debe hacer es ejecutar el comando *terraform init* que descarga las dependencias necesarias e inicializa el repositorio (ver anexo A-2).

A fin de que Terraform disponga de credenciales para poder aprovisionar recursos es necesario disponer del cliente para interprete de comandos de Azure, *azure-cli* (https://docs.microsoft.com/es-es/cli/azure/install-azure-cli). Una vez se dispone de dicha herramienta se podrá ejecutar el comando *az login* que abrirá un navegador en el que se podrán introducir las credenciales de la cuenta de Azure (ver anexo A-2).

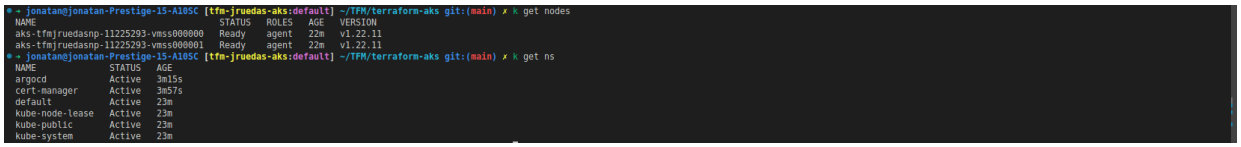
A continuación, se puede generar el plan de Terraform que nos indica que recursos va a crear antes de hacerlo. El comando para generar el plan es *terraform plan -out plan.tf*. De este modo se puede comprobar que se crean los recursos especificados de manera correcta. Cabe destacar que se especifica que el plan se almacena en un fichero llamado *plan.tf* ya que permitirá que el siguiente comando ejecute exactamente el plan generado (ver anexo A-4).

Para finalizar, se ejecuta el comando que utilizará el plan generado para aprovisionar los recursos necesarios en Azure. Dicho comando es *terraform apply "plan.tf"*. Terraform leerá el

plan generado que está almacenado en el fichero *plan.tf* y empezará el aprovisionamiento de los recursos de manera ordenada (ver anexo A-5).

Una vez finalizado el aprovisionamiento se podrá acceder al clúster desde la línea de comandos ya que el fichero de configuración *kubeconfig* se ha establecido como el contexto actual.

Figura 37. *Listado de nodos y namespaces.*



Fuente. Elaboración propia.

4.4.2.5. Configuración de ArgoCD

Para llevar a cabo el despliegue de las aplicaciones se ha utilizado el patrón *App of Apps* (ArgoCD, 2022b) que consiste en crear una aplicación que englobe al resto de aplicaciones que se quieren desplegar. De esta manera, inicialmente, ArgoCD despliega la aplicación principal, la cual tendrá la configuración para que ArgoCD despliegue el resto de las aplicaciones que se necesiten. Gracias a esto se consigue un despliegue global de manera desatendida de todo un clúster junto con las aplicaciones deseadas.

Además, las aplicaciones desplegadas por la aplicación principal también pueden auto gestionarse o gestionarse por separado por lo que se consigue una gran flexibilidad a la hora de inicializar un flujo de GitOps.

El repositorio de configuración se ha dividido en las siguientes carpetas:

- **Applications.** Esta carpeta es la carpeta raíz de la aplicación padre que se encarga de desplegar el resto de las aplicaciones. A pesar de no ser una aplicación real, esta carpeta debe seguir la estructura definida por Helm. Por consiguiente, dentro se puede encontrar lo siguiente:
    - **Templates.** Es una carpeta que contiene las definiciones de las aplicaciones que se instalarán en el clúster.
    - **Chart.yaml.** Es el fichero que contiene los metadatos del *Chart* de la aplicación raíz. Recordemos que se utiliza este formato para que ArgoCD pueda gestionarlo fácilmente.

- **Values.yaml.** Es el fichero de valores que se podría utilizar para emplantillar las aplicaciones definidas en la carpeta *templates*.

- **Charts.** Contiene los *Charts* de las aplicaciones. En el caso de la aplicación de demostración es la configuración de esta. Sin embargo, el resto de los *Charts* son envoltorios de los oficiales. Esto permite parametrizarlos de manera sencilla mediante su fichero de *values.yaml*.

A continuación, se procede a explicar las configuraciones de las distintas aplicaciones.

Aplicación raíz

Tal y como pudo verse en el fichero de configuración utilizado en el proyecto de Terraform para instalar ArgoCD, ver Figura 36, la aplicación raíz se instala como aplicación adicional tras instalar ArgoCD. Esta aplicación define el repositorio de origen, y la carpeta dentro de ese directorio, donde se encuentran las definiciones de las aplicaciones de ArgoCD para que este pueda instalarlas.

Además, se establece como clúster de destino el mismo en el que está ejecutando ArgoCD. Cabe destacar que podría ser un clúster remoto, lo que facilita la separación de los entornos de desarrollo de los de producción.

Por último, se define unas políticas de sincronización y reintentos en caso de que se deba reintentar por algún fallo temporal.

Además, existe un fichero *values.yaml* que permitiría reemplazar valores si fuese necesario.

Figura 38. *Aplicación raíz.*

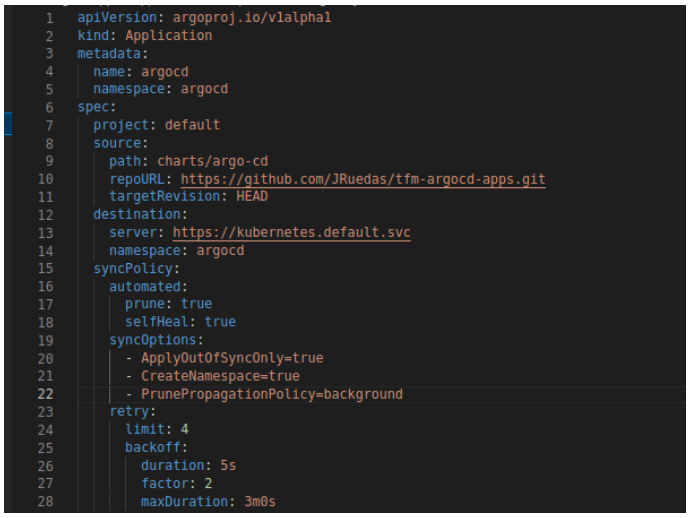
```
23  additionalApplications:
24    - name: root-app
25      namespace: argocd
26      project: default
27      source:
28        path: applications/
29        repoURL: https://github.com/JRuedas/tfm-argocd-apps.git
30        targetRevision: HEAD
31      destination:
32        server: https://kubernetes.default.svc
33        namespace: root-app
34      syncPolicy:
35        automated:
36          prune: true
37          selfHeal: true
38        syncOptions:
39          - CreateNamespace=true
40          - ApplyOutOfSyncOnly=true
41          - PrunePropagationPolicy=background
42        retry:
43          limit: 4
44          backoff:
45            duration: 5s
46            factor: 2
47            maxDuration: 3m0s
```

Fuente. Elaboración propia.

ArgoCD

La configuración utiliza el mismo repositorio que la aplicación raíz, el repositorio de configuración. Sin embargo, la carpeta dentro de este repositorio es distinta ya que hace referencia al *Chart* que se ha creado. Cabe destacar que el *namespace* de destino es el mismo que se configuró en Terraform, *argocd.* Esta figura muestra la definición de la aplicación.

Figura 39. *Aplicación de ArgoCD.*

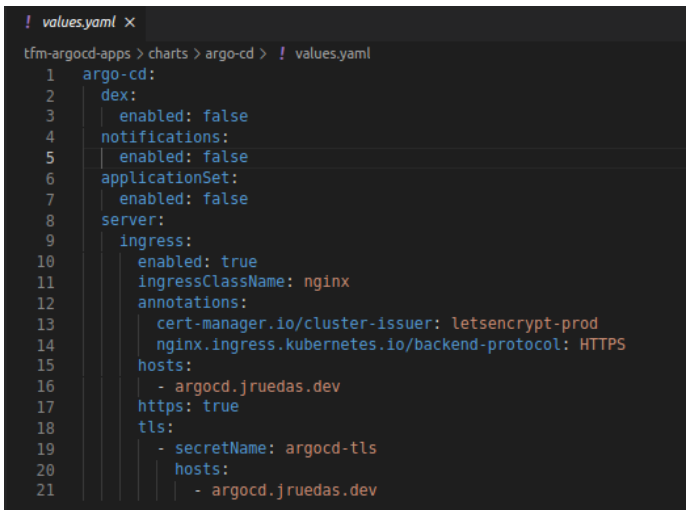


```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: argocd
  namespace: argocd
spec:
  project: default
  source:
    path: charts/argo-cd
    repoURL: https://github.com/JRuedas/tfm-argocd-apps.git
    targetRevision: HEAD
  destination:
    server: https://kubernetes.default.svc
    namespace: argocd
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - ApplyOutOfSyncOnly=true
      - CreateNamespace=true
      - PrunePropagationPolicy=background
    retry:
      limit: 4
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m0s
```

Fuente. Elaboración propia.

Se han deshabilitado características que no eran necesarias para aligerar al máximo la aplicación y se ha configurado el *Ingress* para poder acceder mediante el dominio *argocd.jruedas.dev* usando HTTPS al panel de administración de ArgoCD.

Figura 40. *Valores por defecto para ArgoCD.*



values.yaml

tfm-argocd-apps > charts > argo-cd > values.yaml

```yaml
argo-cd:
  dex:
    enabled: false
  notifications:
    enabled: false
  applicationSet:
    enabled: false
  server:
    ingress:
      enabled: true
      ingressClassName: nginx
      annotations:
        cert-manager.io/cluster-issuer: letsencrypt-prod
        nginx.ingress.kubernetes.io/backend-protocol: HTTPS
      hosts:
        - argocd.jruedas.dev
      https: true
      tls:
        - secretName: argocd-tls
          hosts:
            - argocd.jruedas.dev
```

Fuente. Elaboración propia.

Nginx Ingress Controller

En esta aplicación se cambia la carpeta dentro del repositorio origen y el *namespace* del destino, que en este caso es *ingress-nginx*. El resto de los valores se mantienen como en el resto de las aplicaciones definidas.

Figura 41. *Aplicación de Ingress Nginx.*

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: ingress-nginx
  namespace: argocd
spec:
  project: default
  source:
    path: charts/ingress-nginx
    repoURL: https://github.com/JRuedas/tfm-argocd-apps.git
    targetRevision: HEAD
  destination:
    server: https://kubernetes.default.svc
    namespace: ingress-nginx
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - ApplyOutOfSyncOnly=true
      - CreateNamespace=true
      - PrunePropagationPolicy=background
    retry:
      limit: 3
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m0s
```

Fuente. Elaboración propia.

No ha sido necesario definir ningún tipo de valor para esta aplicación, se han utilizado los valores por defecto del *Chart*.

Cert Manager

El patrón se repite, al igual que en el resto de las aplicaciones, cambiando la carpeta del repositorio de origen y el *namespace* de destino que en este caso es *cert-manager*. Sin embargo, esta aplicación destaca porque tiene un campo nombrado como *ignoreDifferences*.

Este campo se ha establecido para evitar que ArgoCD compruebe las diferencias de dicho recurso ya que por un fallo que existe en la versión nunca llega a sincronizarse correctamente lo que causa que ArgoCD entre en un bucle de sincronización. Ya hay planeada una actualización que arregla este problema, pero aún no ha sido liberada.

Figura 42. *Aplicación de Cert manager.*

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: cert-manager
  namespace: argocd
spec:
  project: default
  source:
    path: charts/cert-manager
    repoURL: https://github.com/JRuedas/tfm-argocd-apps.git
    targetRevision: HEAD
  destination:
    server: https://kubernetes.default.svc
    namespace: cert-manager
  ignoreDifferences:
  - group: admissionregistration.k8s.io
    kind: ValidatingWebhookConfiguration
    name: cert-manager-webhook
    jqPathExpressions:
      - .webhooks[].namespaceSelector.matchExpressions[] | select(.key == "control-plane")
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - ApplyOutOfSyncOnly=true
      - CreateNamespace=true
      - PrunePropagationPolicy=background
    retry:
      limit: 3
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m0s
```

Fuente. Elaboración propia.

Respecto a la configuración de la aplicación, este caso únicamente necesitaba que los CRD (Custom Resource Definitions) Kubernetes se instalasen junto al *Chart*.

Figura 43. *Valores por defecto para Cert manager.*

```yaml
cert-manager:
  installCRDs: true
```

Fuente. Elaboración propia.

External DNS

Para External DNS se ha definido la siguiente aplicación (ver Figura 44). Como puede verse, de nuevo, el *namespace* (*external-dns*) del destino y la carpeta de origen es lo único que cambia.

Los valores definidos para esta aplicación, ver Figura 45, permiten manejar las zonas DNS de Azure. Por lo tanto, se deben establecer una serie de parámetros de configuración que indican que el proveedor Azure, que el nombre de dominio es *jruedas.dev*, y la localización de las credenciales para poder utilizar el API de Azure.

Estas credenciales se inyectan en un secreto del clúster mediante la aplicación External Secrets y esta configuración monta el secreto como volumen en la ruta */etc/kubernetes* para que la aplicación pueda utilizarlas.

Figura 44. *Aplicación de External DNS.*

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: external-dns
  namespace: argocd
spec:
  project: default
  source:
    path: charts/external-dns
    repoURL: https://github.com/JRuedas/tfm-argocd-apps.git
    targetRevision: HEAD
  destination:
    server: https://kubernetes.default.svc
    namespace: external-dns
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - ApplyOutOfSyncOnly=true
      - CreateNamespace=true
      - PrunePropagationPolicy=background
    retry:
      limit: 3
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m0s
```

Fuente. Elaboración propia.

Figura 45. *Valores por defecto para External DNS.*

```yaml
external-dns:
  domainFilters:
    - jruedas.dev
  provider: azure
  txtPrefix: externaldns-
  extraArgs:
    - --azure-resource-group=tfm-jruedas-rg
  extraVolumes:
    - name: azure-config-file
      secret:
        secretName: azure-config-file
  extraVolumeMounts:
    - name: azure-config-file
      mountPath: /etc/kubernetes
      readOnly: true
```

Fuente. Elaboración propia.

External Secrets

Es la aplicación encargada de conectarse a Azure Key Vault para descargar los secretos almacenados en la bóveda creada por Terraform.

Figura 46. *Aplicación de External Secrets.*

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: external-secrets
  namespace: argocd
spec:
  project: default
  source:
    path: charts/external-secrets
    repoURL: https://github.com/JRuedas/tfm-argocd-apps.git
    targetRevision: HEAD
  destination:
    server: https://kubernetes.default.svc
    namespace: external-secrets
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - ApplyOutOfSyncOnly=true
      - CreateNamespace=true
      - PrunePropagationPolicy=background
    retry:
      limit: 3
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m0s
```

Fuente. Elaboración propia.

El principal cambio respecto al resto de aplicaciones es la carpeta dentro del repositorio de origen y el *namespace* dentro del clúster de destino, que en este caso es *external-secrets*.

Para esta aplicación se utilizan los valores por defecto. Sin embargo, el directorio *templates* del *Chart* contiene dos ficheros que permiten crear los recursos para recuperar los secretos de Azure. Estos recursos son el *ClusterSecretKeyStore* (ver Figura 47) y el Cluster*ExternalSecret* (ver Figura 48).

El primero hace referencia a la bóveda que se ha aprovisionado en la nube y la política de acceso a la misma, mientras que el segundo es el encargado de recuperar los secretos para inyectarlos en el clúster de Kubernetes y de este modo hacerlos accesibles para que las aplicaciones puedan utilizarlos. En este caso, se recupera el secreto con las credenciales para que la aplicación External DNS pueda manejar las zonas DNS del proveedor.

Figura 47. *Cluster Secret Store de External Secrets.*

```yaml
apiVersion: external-secrets.io/v1beta1
kind: ClusterSecretStore
metadata:
  name: jruedas-az-secret-store
spec:
  provider:
    azurekv:
      authType: ManagedIdentity
      vaultUrl: https://jruedas-tfm-vault.vault.azure.net
```

Fuente. Elaboración propia.

Figura 48: *Cluster External Secret de External Secrets.*

```yaml
apiVersion: external-secrets.io/v1beta1
kind: ClusterExternalSecret
metadata:
  name: jruedas-az-secret
spec:
  namespaceSelector:
    matchLabels:
      kubernetes.io/metadata.name: external-dns
  refreshTime: 1m
  externalSecretSpec:
    refreshInterval: 0h
    secretStoreRef:
      kind: ClusterSecretStore
      name: jruedas-az-secret-store
    target:
      name: azure-config-file
      creationPolicy: Owner
    data:
    - secretKey: azure.json
      remoteRef:
        key: external-dns-secret
```

Fuente. Elaboración propia.

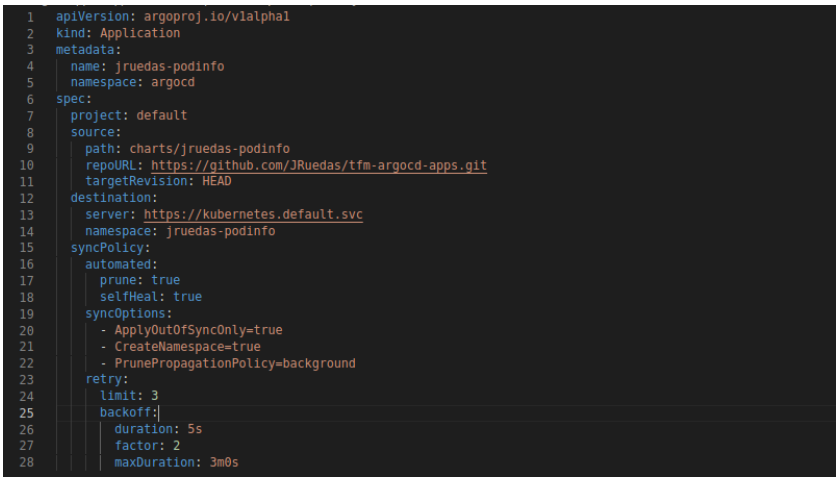
JRuedas-Podinfo

Esta aplicación es la que se ha escogido para hacer la demostración. Las otras aplicaciones se instalan en el clúster como un apoyo para que esta aplicación pueda funcionar como si de un clúster de producción se tratase. Por ese motivo, el resto de las aplicaciones se han definido

como un envoltorio de los *Charts* oficiales mientras que esta sí que dispone de los ficheros de YAML que definen los recursos que se instalarán en el clúster.

Como puede verse, la diferencia respecto a las otras aplicaciones sigue siendo la misma, la carpeta de origen y el *namespace* de destino del clúster, "jruedas-podinfo" en este caso.

Figura 49. *Aplicación de Podinfo.*

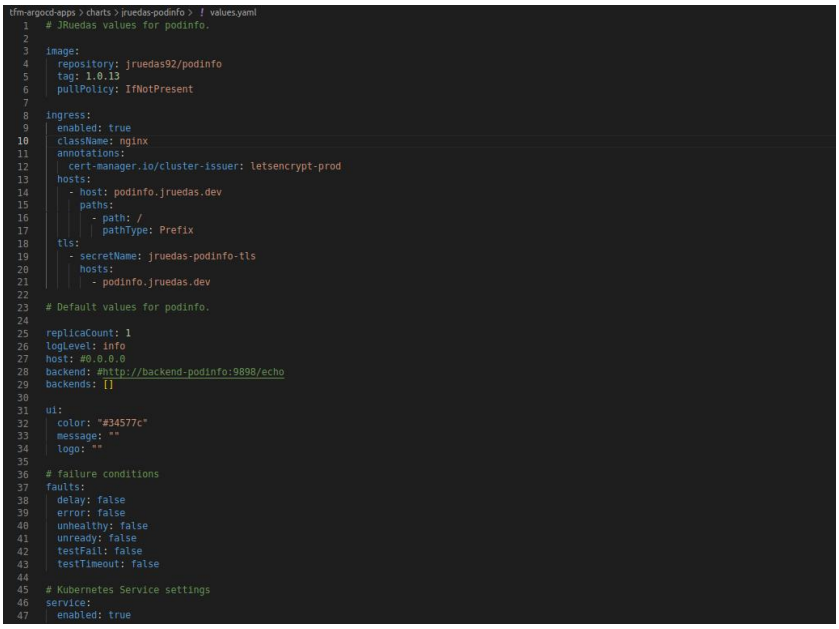


Fuente. Elaboración propia.

La Figura 50 muestra los valores por defecto. Existen dos secciones delimitadas por un comentario. La primera sección define los valores que se usan para este trabajo, en concreto son valores relacionados con la versión de la imagen y la configuración de *Ingress*. La siguiente sección, línea veintitrés, define los valores por defecto que traía la aplicación de demostración y que no han sido cambiados para esta demostración. Como puede verse, la configuración del *Ingress* permitirá acceder mediante el dominio *podinfo.jruedas.dev* utilizando el protocolo HTTPS.

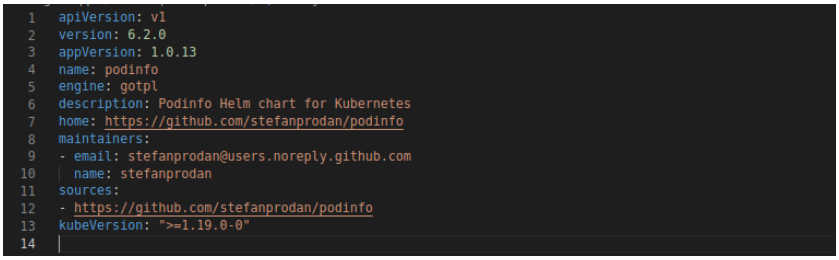
Al generar una nueva versión el código fuente, el parámetro *tag* que se encuentra dentro del fichero en la sección *image* cambiará a la nueva versión y se aplicará. Además, también cambiará el valor del parámetro *appVersion* que se encuentra dentro del fichero *Chart.yaml* y que define los metadatos del *Chart* (ver Figura 51).

Figura 50. *Valores por defecto para Podinfo.*



Fuente. Elaboración propia.

Figura 51. *Fichero Chart.yaml de Podinfo.*



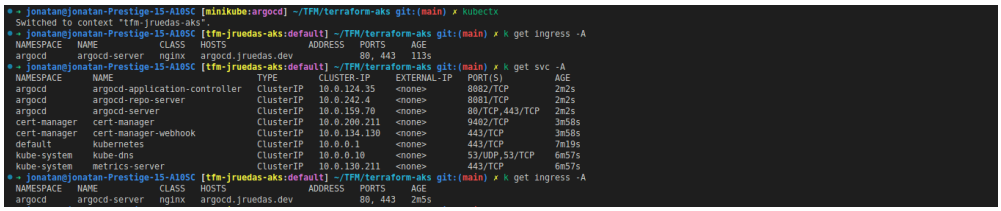
Fuente. Elaboración propia.

4.4.2.6. Ejecución del flujo

Una vez ejecutado Terraform, el clúster ya tiene instalado ArgoCD con la aplicación raíz por lo que el proceso de GitOps se pone en marcha pasado un pequeño periodo de tiempo sin ningún tipo de intervención por parte del operador.

En la Figura 52 se muestra el estado inicial del clúster tras llevar a cabo el aprovisionamiento. Puede verse como ArgoCD ya está instalado junto a Cert manager y es en este momento cuando empieza el proceso de GitOps ya que ArgoCD desplegará la aplicación raíz y entonces empezará a sincronizar el resto de las aplicaciones.

Figura 52. *Estado inicial del clúster.*

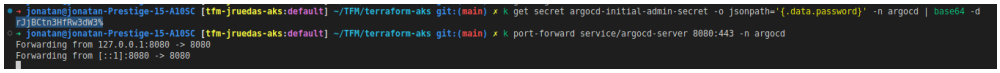


Fuente. Elaboración propia.

Cabe destacar que como el resto de las aplicaciones aún no están instaladas, aunque exista un *Ingress* para ArgoCD, todavía no se puede acceder al panel de control a través del dominio público. Por este motivo se debe acceder mediante un reenvío de puertos desde el *Pod* a la máquina local.

Por último, para poder iniciar sesión en el servidor de ArgoCD se debe recuperar la contraseña de administración que es autogenerada durante la instalación y almacenada en un secreto. Es necesario decodificar la contraseña ya que se encuentra codificada en base 64 (ver Figura 53.

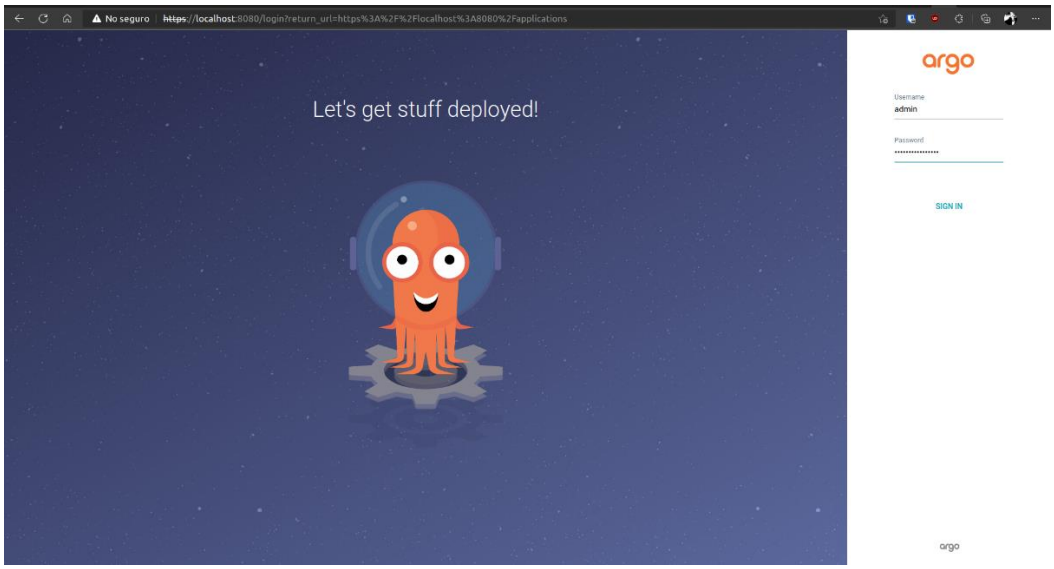
Figura 53. *Contraseña de administrador y reenvío de puertos.*



Fuente. Elaboración propia.

Llegados a este punto, se puede ir a *localhost:8080* para visualizar la página de inicio de sesión de ArgoCD e iniciar sesión con el usuario *admin* y la contraseña recuperada (ver Figura 54).

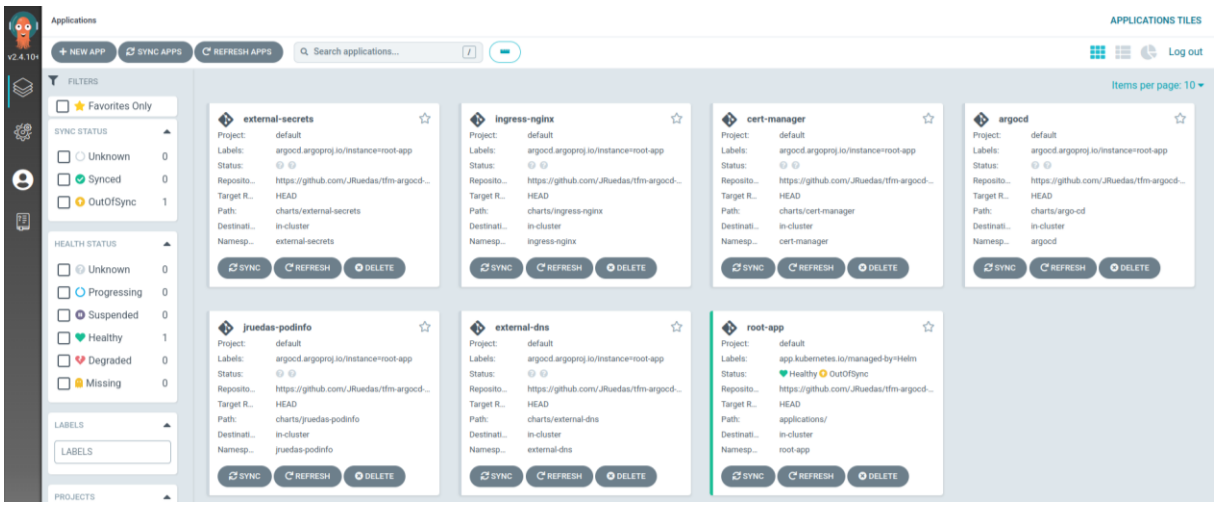
Figura 54. *Página de inicio de sesión de ArgoCD.*



Fuente. ArgoCD. Elaboración propia.

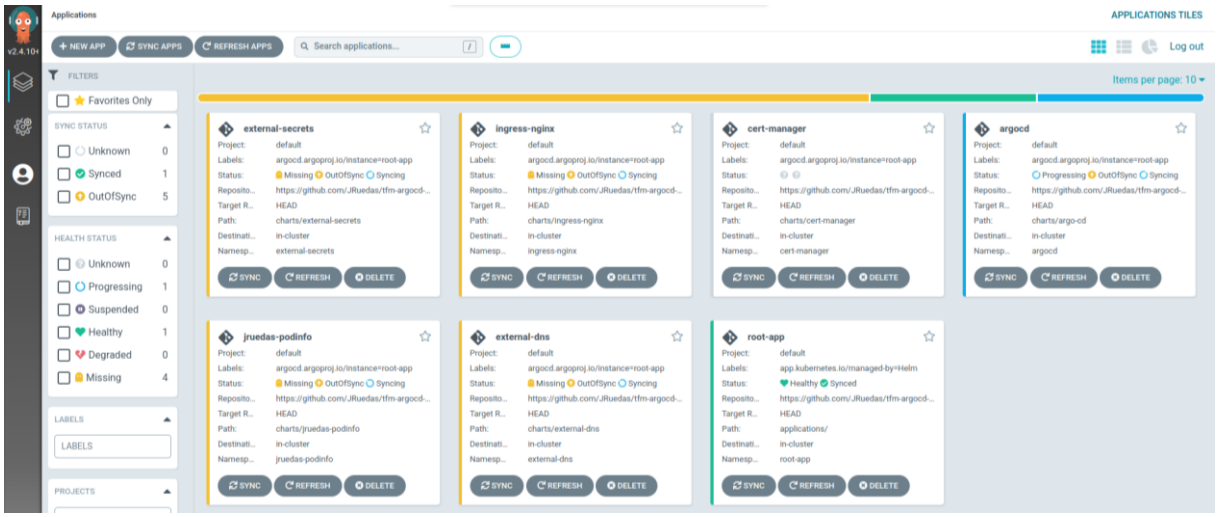
Una vez se haya iniciado sesión podrá verse el panel de control que nos indica las aplicaciones que maneja ArgoCD y su estado. Desde este panel puede verse el estado de sincronización de la aplicación respecto al estado definido en el repositorio de configuración y el estado de salud de esta. Este estado de sincronización tarda en converger un par de minutos, pero eventualmente todas las aplicaciones pasarán a un estado saludable y de sincronización correcto. Las Figura 55, Figura 56, Figura 57 y Figura 58 muestran la evolución del estado.

Figura 55. *Estado de sincronía 1. Inicial.*



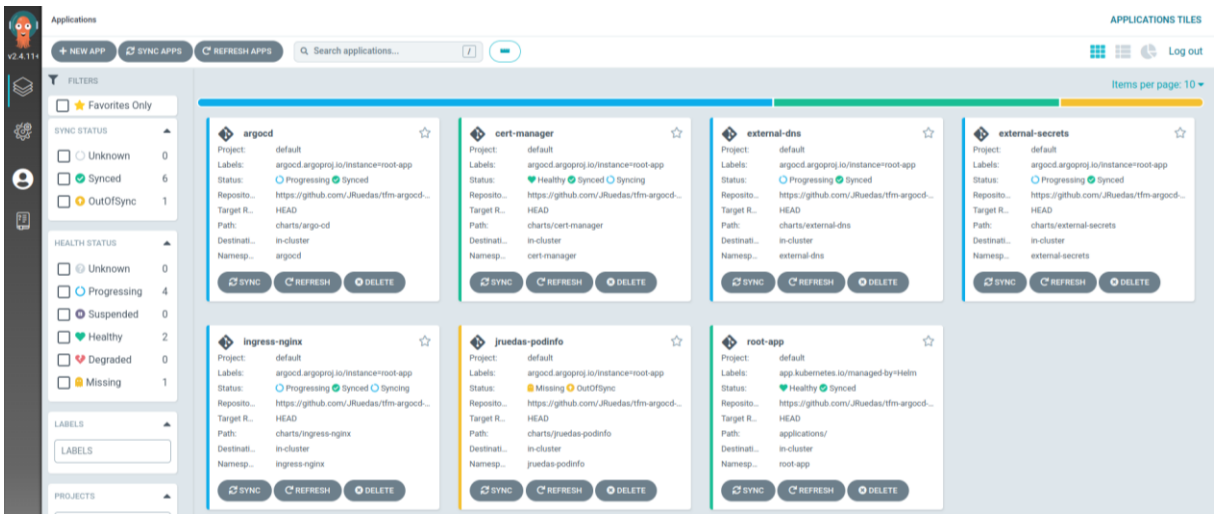
Fuente. ArgoCD. Elaboración propia.

Figura 56. *Estado de sincronía 2. Sincronizando.*



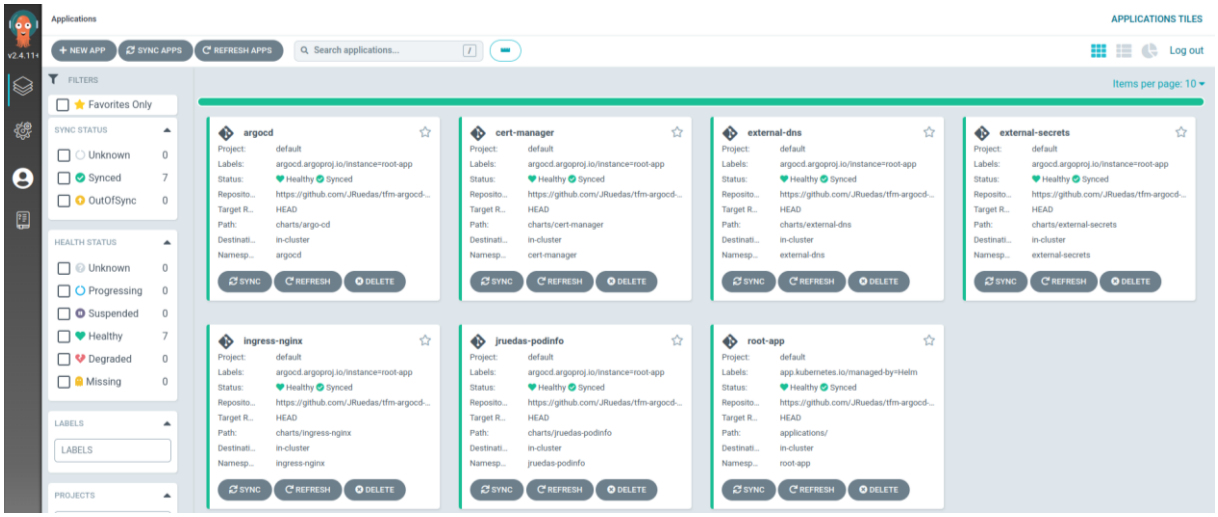
Fuente. ArgoCD. Elaboración propia.

Figura 57. *Estado de sincronía 3. Sincronizando. Avanzado.*



Fuente. ArgoCD. Elaboración propia.

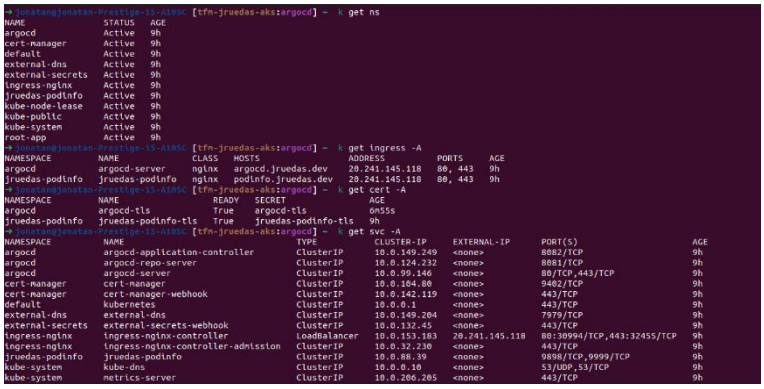
Figura 58. *Estado de sincronía 4. Final. Saludables y sincronizadas.*



Fuente. ArgoCD. Elaboración propia.

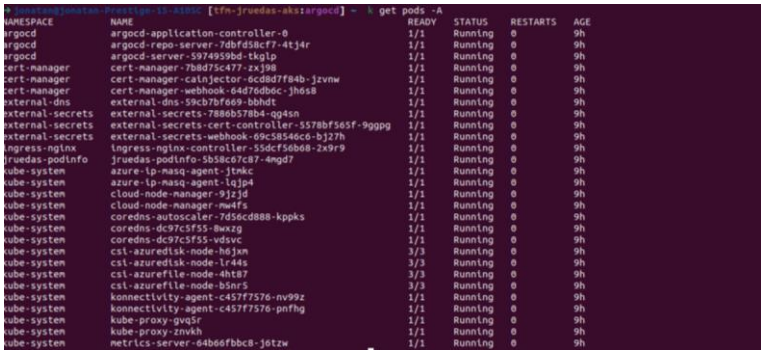
La muestra el estado del clúster una vez que ArgoCD ha conseguido sincronizar los recursos y las aplicaciones se han desplegado.

Figura 59. *Estado del clúster tras sincronizar. 1.*



Fuente. Elaboración propia.

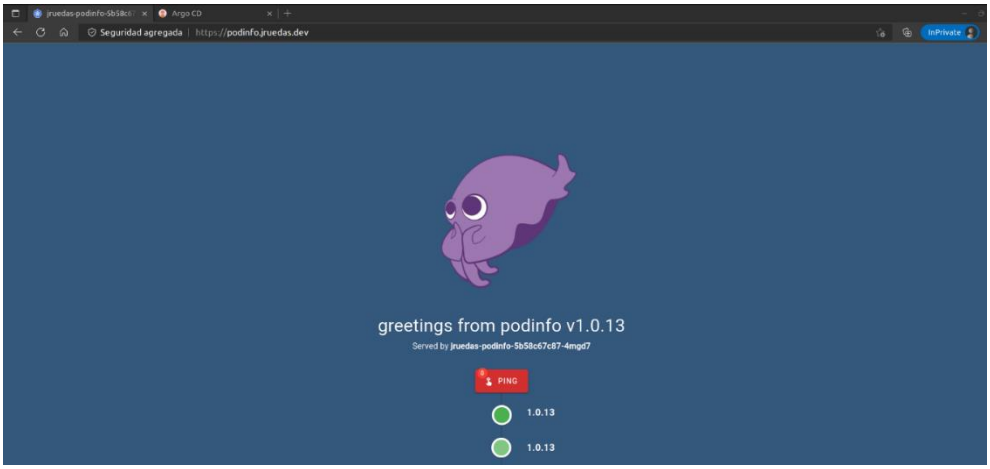
Figura 60. *Estado del clúster tras sincronizar. 2.*



Fuente. Elaboración propia.

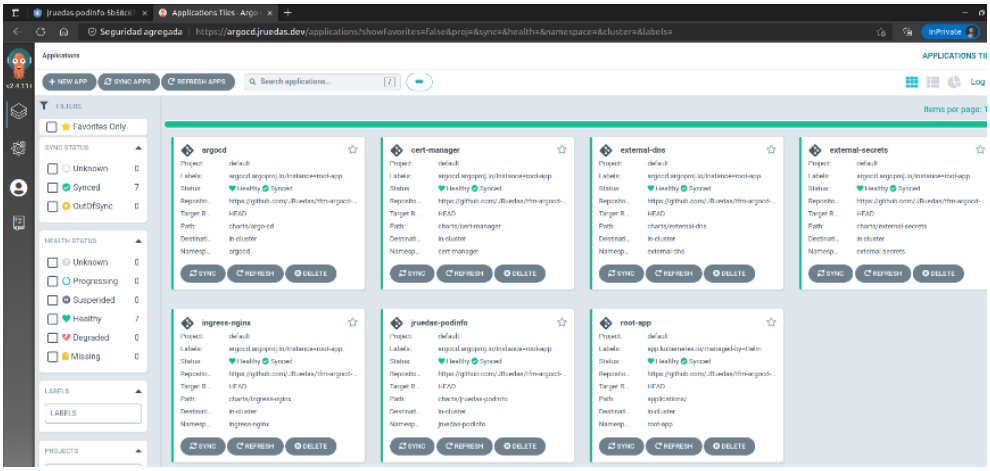
A continuación, se muestra el acceso por HTTPS a las páginas web de ArgoCD y de Podinfo que han sido desplegadas y que disponen de un certificado válido emitido por la autoridad de certificación Let's encrypt (ver Figura 61 y Figura 62).

Figura 61. *Podinfo dominio público.*



Fuente. Podinfo. Elaboración propia.

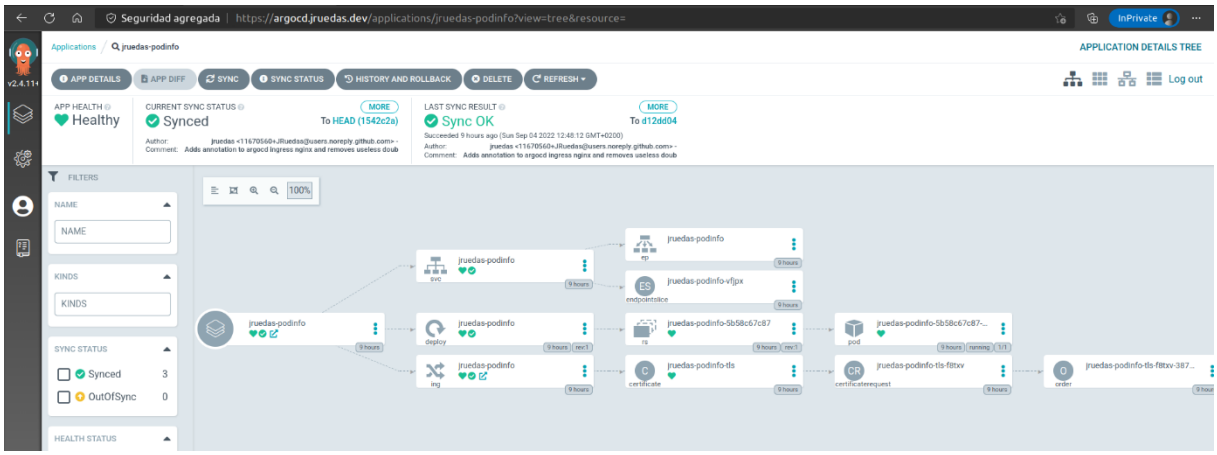
Figura 62. *ArgoCD dominio público.*



Fuente. ArgoCD. Elaboración propia.

Como puede comprobarse, en la Figura 63, al acceder a las distintas aplicaciones ArgoCD muestra el mapa de los recursos que ha creado y las dependencias de estos recursos. Gracias a esto se consigue tener una perspectiva de todos los recursos del recurso que existen en el clúster.

Figura 63. *Recursos de Podinfo en el clúster.*



Fuente. ArgoCD. Elaboración propia.

A hora de liberar una nueva versión de la aplicación bastará con fusionar una *Pull Request* en el repositorio de código fuente para que se ejecute el flujo de trabajo de GitHub Actions que acabará generando los artefactos y actualizando la configuración con la nueva versión. En ese momento, ArgoCD detectará el cambio que se ha realizado en la fuente de verdad y se encargará de reconciliar el estado por lo que la nueva versión se desplegará de manera totalmente automática y transparente para el equipo de operaciones. Ver ejemplo en el anexo C-1.

Además, en caso de borrar algún elemento de la aplicación, ArgoCD verificará que el estado no es el adecuado y recuperará el estado de la aplicación que está definido en Git corrigiendo de esta manera las derivas de configuración que puedan ocurrir eventualmente. Ver ejemplo en el anexo C-2.

4.4.3. Problemas encontrados

Los problemas encontrados han sido variados, pero principalmente se deben a limitaciones de ciertos componentes.

El principal de los problemas ha tenido que ver con que la aplicación Cert manager no permite configurar el recurso *ClusterIssuer* encargado de comunicarse con Let's Encrypt utilizando secretos de Kubernetes. Este recurso tiene que disponer de unas credenciales de Azure ya que necesita poder añadir un registro TXT a la zona DNS para que la entidad certificadora pueda validar a quien pertenece el dominio. El resolutor DNS de Route53 sí que tiene dicha implementación de manera que se evita tener las credenciales hardcodeadas en el fichero de

YAML. En un entorno de GitOps esto es inaceptable por lo que para solventar ese problema se decidió utilizar un módulo de Terraform que instala la aplicación y permite configurar el recurso *ClusterIssuer* utilizando las variables.

La característica mencionada ha sido solicitada mediante la apertura de una incidencia (https://github.com/cert-manager/cert-manager/issues/5412) en el repositorio oficial de GitHub de Cert manager a fin de que esta aplicación siga los principios de GitOps.

Otro de los problemas encontrados ha sido la creación de los recursos necesarios para permitir que la aplicación External Secrets pudiese operar con la bóveda de Azure Key Vault ya que no conseguía acceder a los objetos almacenados. Esto se solucionó mediante la creación de dos políticas de acceso, una de administrador para que el usuario pueda operar mediante la web y otra que únicamente tiene permisos para recuperar los secretos.

Por último, destaco la tarea de documentación sobre el funcionamiento de la autenticación para trabajar con Azure desde Kubernetes ya que no es tan sencillo como en AWS debido a que disponen de múltiples métodos que he tenido que investigar para decidir cual usar.

4.5. Evaluación

El objetivo de este apartado es el de validar si el marco operacional GitOps y el flujo implementado puede aplicarse a un entorno corporativo en el que se trabaja con aplicaciones de Kubernetes.

Uno de los motivos principales que me llevaron a escoger ArgoCD frente a tus competidores es el panel de control gráfico que proporciona. Otras herramientas no proporcionan un elemento tan visual que permita ver rápidamente el estado de las aplicaciones que viven en el clúster.

Se debe destacar que ArgoCD dispone de una gran cantidad de herramientas que permiten conectarlo con herramientas como el correo electrónico, o de mensajería instantánea como Slack, para el envío de notificaciones cuando ocurren eventos relacionados con el estado de salud o de sincronización de las aplicaciones.

Creo que tras llevar a cabo el estudio de GitOps y su implementación, el marco operacional está lo suficientemente maduro como para ser implementado en la empresa de manera gradual. Si bien es cierto que existen muchas variables que pueden dificultar el proceso como

el soporte que puedan proporcionar los propios *Chart* de terceros que se despliegan y la gestión de las credenciales, creo que su implementación es factible sin llevar a cabo una gran cantidad de cambios.

El mayor de los cambios que debe llevarse a cabo es la separación de los repositorios de código fuente y configuración y los cambios asociados a los *pipelines* que tenemos de Jenkins ya que como se ha visto ahora tienen que incrementar la versión de dos repositorios.

Una vez ajustados los *pipelines* y configurados los repositorios lo más importante es detectar que credenciales deben eliminarse de los manifiestos y buscar una alternativa para inyectarlas directamente en el clúster. En este sentido, están surgiendo una gran cantidad de herramientas que intentan solucionar este problema de muchas maneras distintas desde cifrar los secretos con cifrado asimétrico para que solo Kubernetes pueda leerlo hasta inyectarlos de manera externa tal y como se ha llevado a cabo en este trabajo.

Por último, hay que destacar que gracias a este marco operacional se consigue un alto grado de reproducibilidad y replicabilidad que permite crear y destruir entornos nuevos sin mucha dificultad y consiguiendo siempre el mismo resultado.

5. Conclusiones y trabajo futuro

En este apartado se agrupan las conclusiones tanto técnicas como personales que se han obtenido tras la realización de este Trabajo de Fin de Máster.

5.1. Conclusiones

El objetivo principal de este trabajo era el de investigar el funcionamiento de GitOps, sus beneficios y desventajas y llevar a cabo una pequeña evaluación a fin de implementar este marco operacional en la empresa para conseguir que los equipos de operaciones puedan centrarse en tareas que realmente aporten valor a la empresa.

Desde el punto de vista técnico, no es trivial llevar a cabo una implementación de GitOps por muchas razones entre las que se encuentran:

- Desconocimiento del flujo de trabajo de los desarrolladores.
- Limitaciones de las aplicaciones y de los entornos.
- Tiempo invertido en llevar a cabo la implementación.

Sin embargo, el hecho de que no sea trivial no significa que no pueda implantarse de manera ordenada mediante la definición de un *roadmap* que permita aplicar los cambios de manera gradual.

La implementación de este marco permite disponer de entornos altamente automatizados y reduce la incertidumbre que pueda existir a la hora de realizar acciones críticas como pueden ser pases a producción. Además, el paradigma de contenedores que ofrece Kubernetes permite desplegar todo tipo de infraestructura siempre que esta esté empaquetada como una imagen de Docker por lo que se pueden migrar desde servidores web hasta bases de datos u otro tipo de aplicativos.

Considero que la solución aportada es válida ya que se ha demostrado como la liberación de versiones se ha automatizado siguiendo los principios que GitOps propone y obteniendo los beneficios de autogestión y resiliencia esperados.

Por último, desde el punto de vista personal considero que he aprendido mucho sobre esta nueva corriente que ha hecho que mejores mis aptitudes profesionales y mi capacidad de autogestión.

5.2. Líneas de trabajo futuro

Existen una serie de cosas que se han quedado en el tintero por falta de tiempo y que creo que son necesarias para conseguir implementar un flujo de GitOps más resiliente.

La primera es la correcta integración de Cert manager con la nube de Azure. En el momento en el que la característica se añada a la herramienta se podrán referenciar credenciales que se crearán utilizando otras aplicaciones. De esta manera se asegura que el proyecto de Terraform únicamente aprovisiona la infraestructura y lleva a cabo la instalación inicial de ArgoCD.

La segunda es la mejora de la observabilidad. En este caso no se han utilizado herramientas que faciliten la observabilidad de las aplicaciones, del proceso de ArgoCD y de otros elementos. Sin embargo, este es un pilar clave de GitOps por lo que es necesario disponer de herramientas que recolecten logs, comprueben el estado de las aplicaciones, sus métricas y generen notificaciones y alertas cuando algún evento importante ocurra.

Para finalizar no se ha llevado a cabo un despliegue multiclúster que ilustre como se puede trasladar este flujo a otros entornos. Aunque se ha explicado el procedimiento que debería seguirse no se ha aprovisionado un clúster que simule el de producción ni se ha parametrizado los ficheros lo suficiente como para llevarlo a cabo. La promoción entre entornos es un punto muy importante para cualquier empresa y el objetivo es el de facilitar el trabajo que lleva a cabo el equipo de operaciones, no crear nuevos problemas que no tenían anteriormente.

Referencias bibliográficas

Allspaw, J., & Hammond, P. (2009). *10+ Deploys Per Day: Dev and Ops Cooperation at Flickr.* Slideshare. https://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr

Amazon Web Services. (2020). *What is DevOps? - Amazon Web Services (AWS).* https://aws.amazon.com/devops/what-is-devops/

AppDynamics. (2022). *BizDevOps: The Evolution from Agile and DevOps.* AppDynamics. https://www.appdynamics.com/solutions/devops/bizdevops/index.html

ArgoCD. (2021). *Declarative Setup - Manage Argo CD Using Argo CD.* ArgoCD Official Documentation. https://argo-cd.readthedocs.io/en/stable/operator-manual/declarative-setup/#manage-argo-cd-using-argo-cd

ArgoCD. (2022a). *Argo CD - Declarative GitOps CD for Kubernetes.* ArgoCD Official Documentation. https://argo-cd.readthedocs.io/

ArgoCD. (2022b). *Custer Bootstraping - Apps of Apps pattern.* ArgoCD official documentation. https://argo-cd.readthedocs.io/en/stable/operator-manual/cluster-bootstrapping/#app-of-apps-pattern

Atlassian. (2021). *What is DevOps?* Atlassian. https://www.atlassian.com/devops

Beetz, F., & Harrer, S. (2021). GitOps: The Evolution of DevOps? *IEEE Software,* 0. 10.1109/MS.2021.3119106

Beetz, F., Kammer, A. & Harrer, S. (2022). *GitOps.* GitOps tech. https://www.gitops.tech/

Branscombe, M. (2019). The Current State of AIOps. https://thenewstack.io/the-current-state-of-aiops/

Brown, D. (2015). What is DevOps? https://www.donovanbrown.com/post/what-is-devops

Buchanan, I. (2020). *History of DevOps.* Atlassian. https://www.atlassian.com/devops/what-is-devops/history-of-devops

Buehring, S. (2021). *What is DevOps?: Free ebook.* Knowledge Train. https://www.knowledgetrain.co.uk/it/devops/devops-training/what-is-devops

Databricks. (2020). *MLOps - Databricks Glossary.* Databricks. https://databricks.com/glossary/mlops

DataKitchen. (2022). *What Is DataOps?* DataKitchen. https://datakitchen.io/what-is-dataops/

DeBoer, E. (2019). *CALMS: A Principle-based DevOps Framework.* Sonatype. https://blog.sonatype.com/principle-based-devops-frameworks-calms

Dhaduk, H. (2022, -01-13). DevOps Lifecycle: 7 Phases Explained in Detail with Examples. https://www.simform.com/blog/devops-lifecycle/

Flux. (2022). *Flux - the GitOps family of projects.* Flux Official Documentation. https://fluxcd.io/

Garfield, D., & Rigby, S. (2021). *OpenGitOps 1.0 is finally here and why you should care.* Open GitOps. https://opengitops.dev/blog/1.0-announcement/

Gartner. (2019). *Definition of DevOps - Gartner Information Technology Glossary.* Gartner. https://www.gartner.com/en/information-technology/glossary/devops

Gartner. (2021). *Definition of AIOps (Artificial Intelligence for IT Operations) - Gartner Information Technology Glossary.* Gartner. https://www.gartner.com/en/information-technology/glossary/aiops-artificial-intelligence-operations

Geertsema, A. (2021). *BizDevOps, aligning business and IT.* Medium. https://arjangeertsema.medium.com/bizdevops-aligning-business-and-it-ea00ada05966

Gitlab. (2022). *What is GitOps? | GitLab.* Gitlab. https://about.gitlab.com/topics/gitops

Hemo, I. B. (2020). *Council Post: Three Reasons Why DataOps Will Boom In 2021.* Forbes. https://www.forbes.com/sites/forbestechcouncil/2020/12/21/three-reasons-why-dataops-will-boom-in-2021/

IBM. (2019). *What is DataOps?* IMB Journey to AI Blog. https://www.ibm.com/blogs/journey-to-ai/2019/12/what-is-dataops/

Ismail, N. (2018). Why DevOps must become BizDevOps for business and IT collaboration. https://www.information-age.com/devops-bizdevops-business-123471568/

Jabbari, R., bin Ali, N., Petersen, K., & Tanveer, B. (May 24, 2016). What is DevOps?: A Systematic Mapping Study on Definitions and Practices. Paper presented at the 1-11. 10.1145/2962695.2962707 http://dl.acm.org/citation.cfm?id=2962707

JenkinsX. (2022). *Jenkins X 3.x.* Jenkins X - Cloud Native CI/CD Built On Kubernetes. https://jenkins-x.io/v3/

Kapelonis, K. (2021). Stop Using Branches for Deploying to Different GitOps Environments. https://codefresh.io/blog/stop-using-branches-deploying-different-gitops-environments/

Kim, G. (2012). The Three Ways: The Principles Underpinning DevOps. https://itrevolution.com/the-three-ways-principles-underpinning-devops/

Kim, G., Behr, K., & Spafford, G. (2018). *The Phoenix Project* (Third edition ed.). IT Revolution.

Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2021). *The DevOps Handbook*. IT Revolution Press.

Koskinen, A. (2019). DevSecOps: Building security into the core of DevOps., 56-59. https://jyx.jyu.fi/handle/123456789/67345

Matthews, K. (2019). What Is AIOps — And Why You Should Care. https://thenewstack.io/what-is-aiops-and-why-you-should-care/

Mendix. (2022). *What is Low-Code? An Introduction to Low-Code Development.* Mendix. https://www.mendix.com/low-code-guide/

Merritt, R. (2020). *What is MLOps?* NVIDIA Blog. https://blogs.nvidia.com/blog/2020/09/03/what-is-mlops/

Microsoft. (2019). *What is DevOps? DevOps Explained.* Microsoft Azure. https://azure.microsoft.com/en-us/overview/what-is-devops/

New Relic. (2018). *What is DevOps - Explained.* New Relic. https://newrelic.com/devops/what-is-devops

ODSC-Open Data Science. (2019). *DataOps and the DataOps Manifesto.* Medium. https://odsc.medium.com/dataops-and-the-dataops-manifesto-fc6169c02398

Peterson, J. (2020). *DevSecOps vs. SecDevOps: A Rose by Any Other Name?* WhiteSource. https://www.whitesourcesoftware.com/resources/blog/devsecops-vs-secdevops/

Putano, B. (2017). *A Quick Guide to BizDevOps.* Stackify. https://stackify.com/bizdevops-guide/

Ruka, A. (2017). *OneFlow - a Git branching model and workflow | End of Line Blog.* End of Line Blog. https://www.endoflineblog.com/oneflow-a-git-branching-model-and-workflow#when-to-use-oneflow

ScienceLogic. (2019). Are you ready for AIOps? Here's what you need to know. https://sciencelogic.com/blog/are-you-ready-for-aiops-what-you-need-to-know

Sharma, N. (2021). *MLOps vs AIOps – What's the Difference?* neptune.ai. http://neptune.ai/blog/mlops-vs-aiops-differences

Singh, R. (2019). *The Most Important Elements of AIOps - DZone AI.* dzone.com. https://dzone.com/articles/the-most-important-elements-of-aiops

Tekton. (2022). *Welcome to Tekton.* Tekton. https://tekton.dev/docs/

TheDataOpsManifesto. (2022). *The DataOps Manifesto.* DataOps Manifesto. https://dataopsmanifesto.org/en/

Weaveworks. (2017). *GitOps - Operations by Pull Request.* https://www.weave.works

Weaveworks. (2018). *What Is GitOps | Weaveworks.* Weaveworks. https://www.weave.works/blog/what-is-gitops-really

Weaveworks. (2021). *Highlights from the 2021 State of DevOps Report.* Weaveworks. https://www.weave.works/highlights-2021-state-of-devops

Weaveworks. (2022). *GitOps what you need to know.* Weaveworks. https://www.weave.workshttps://www.weave.works/technologies/gitops

Webster, E. (2021). What is MLOps? - Benefits, how it works, and DevOps vs. MLOps. https://nealanalytics.com/blog/what-is-mlops/

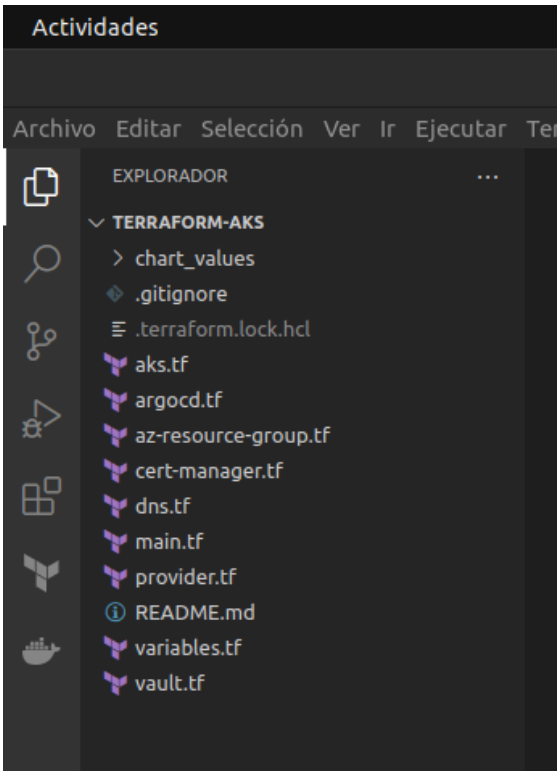
Wiggins, A. (2017). *The Twelve-Factor App.* 12factor. https://12factor.net/

Williams, J. (2018). Beyond “shifting left” – An exploration of DevSecOps. https://jaxenter.com/exploration-devsecops-144849.html

Yuen, B., Matyushentsev, A., Suen, J., & Ekenstam, T. (2021). *GitOps and Kubernetes*

Anexo A. Proyecto de Terraform

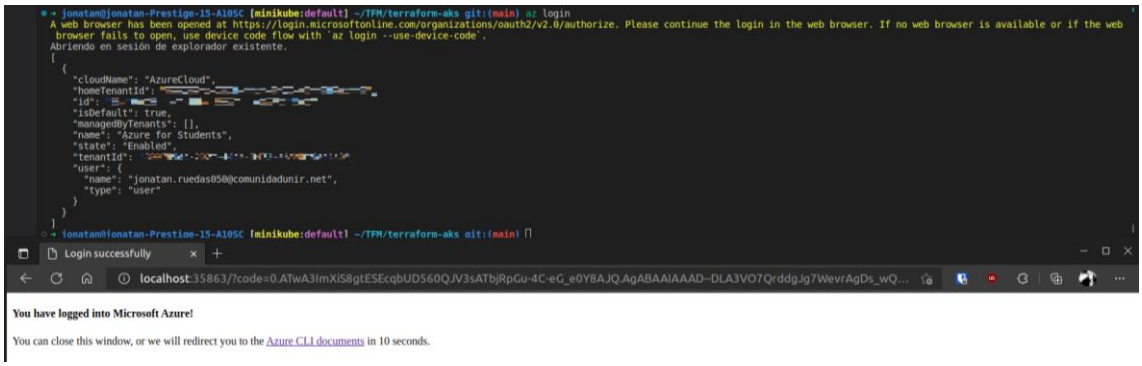
A-1. Estructura del proyecto



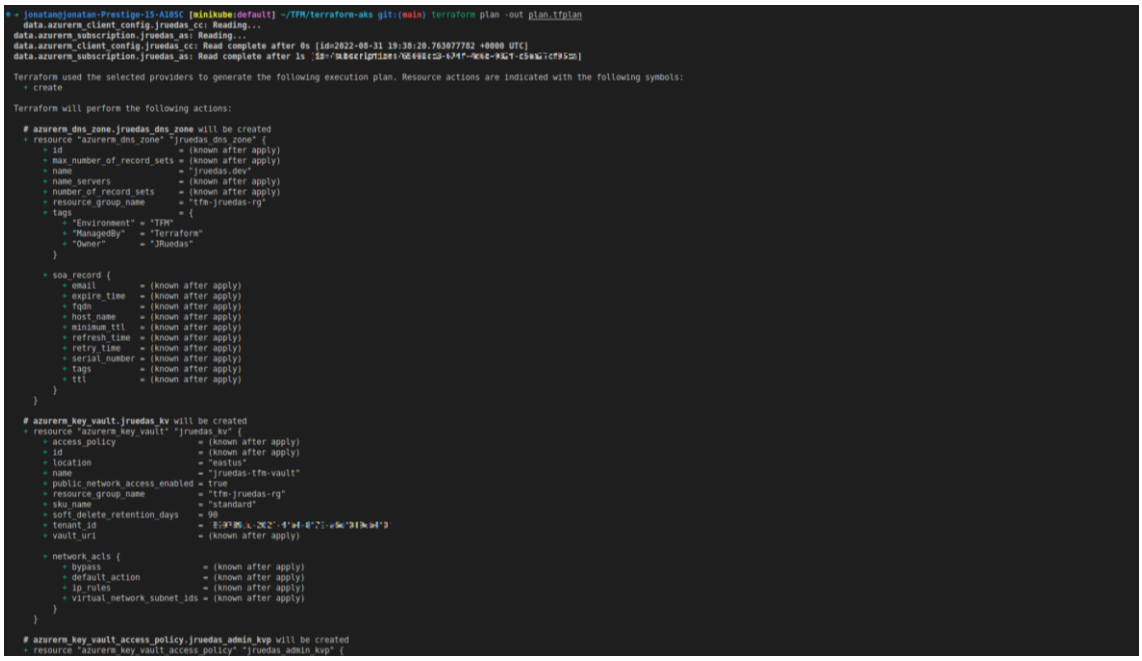
A-2. Inicialización del proyecto

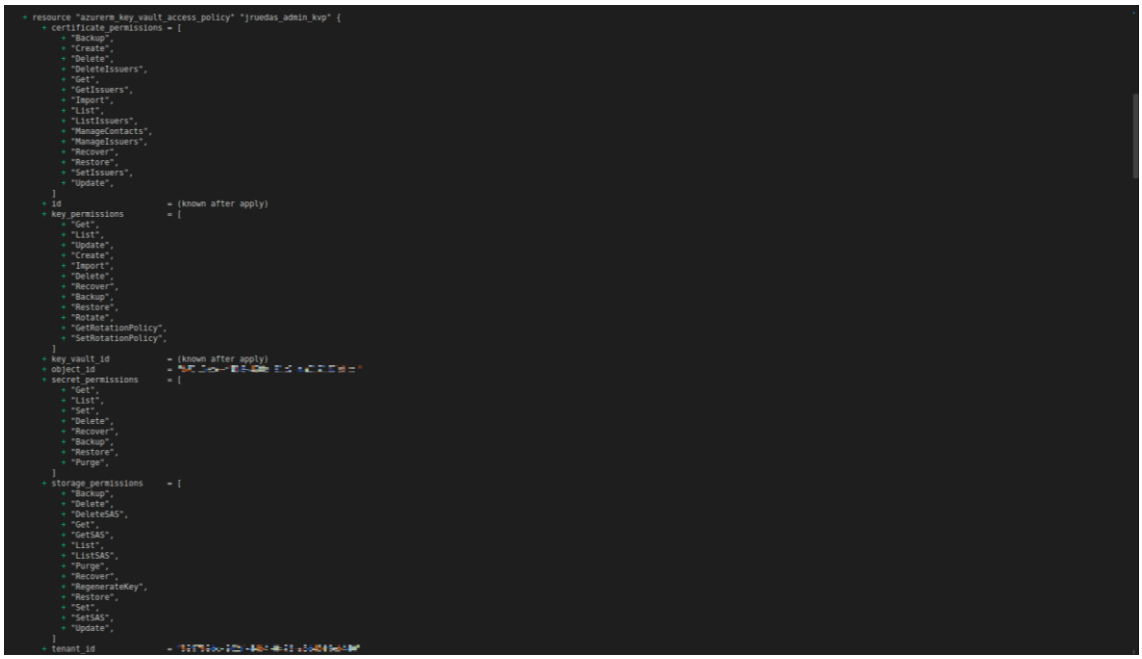
```
jonatan@jonatan-Prestige-15-A10SC [minikube:default] ~/TFM/terraform-aks git:(main) terraform init
Initializing modules...
Downloading registry.terraform.io/terraform-iaac/cert-manager/kubernetes 2.4.2 for cert-manager...
- cert-manager in .terraform/modules/cert-manager
- cert-manager.certificates in .terraform/modules/cert-manager/modules/_certificate

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/helm from the dependency lock file
- Reusing previous version of gavinbunney/kubectl from the dependency lock file
- Reusing previous version of hashicorp/local from the dependency lock file
- Reusing previous version of hashicorp/kubernetes from the dependency lock file
- Reusing previous version of hashicorp/time from the dependency lock file
- Reusing previous version of hashicorp/azurerm from the dependency lock file
- Installing hashicorp/kubernetes v2.13.0...
- Installed hashicorp/kubernetes v2.13.0 (signed by HashiCorp)
- Installing hashicorp/time v0.8.0...
- Installed hashicorp/time v0.8.0 (signed by HashiCorp)
- Installing hashicorp/azurerm v3.19.1...
- Installed hashicorp/azurerm v3.19.1 (signed by HashiCorp)
- Installing hashicorp/helm v2.6.0...
- Installed hashicorp/helm v2.6.0 (signed by HashiCorp)
- Installing gavinbunney/kubectl v1.14.0...
- Installed gavinbunney/kubectl v1.14.0 (self-signed, key ID AD64217B5ADD572F)
- Installing hashicorp/local v2.2.3...
- Installed hashicorp/local v2.2.3 (signed by HashiCorp)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
jonatan@jonatan-Prestige-15-A10SC [minikube:default] ~/TFM/terraform-aks git:(main)
```

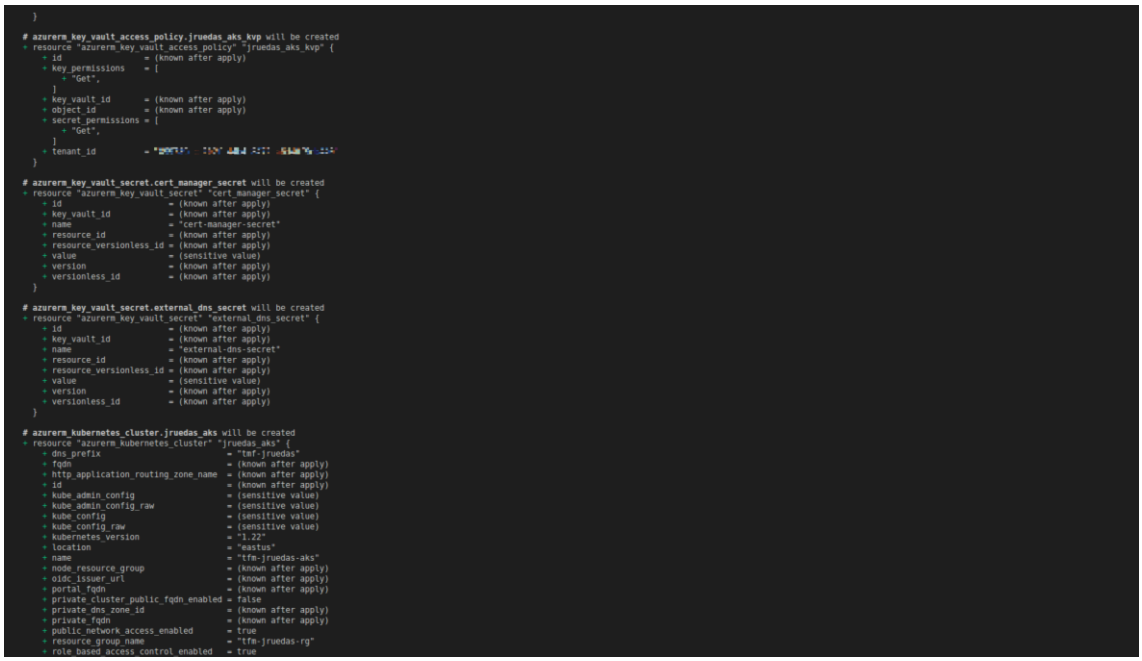
A-3. Inicio de sesión con el cliente de Azure



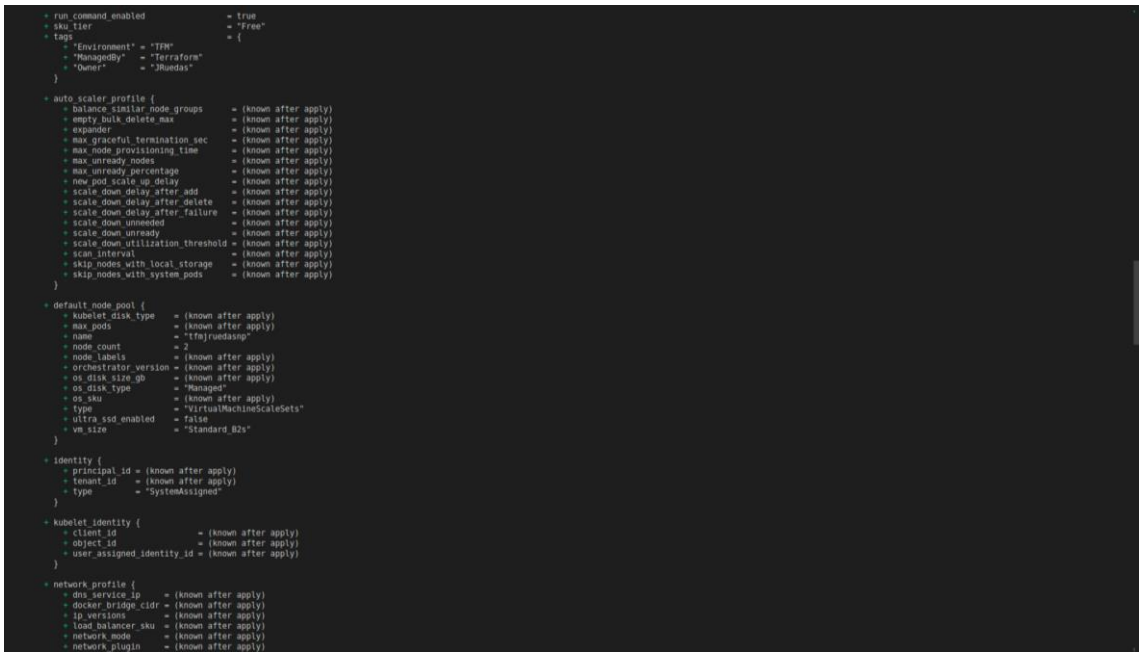
A-4. Creación del plan



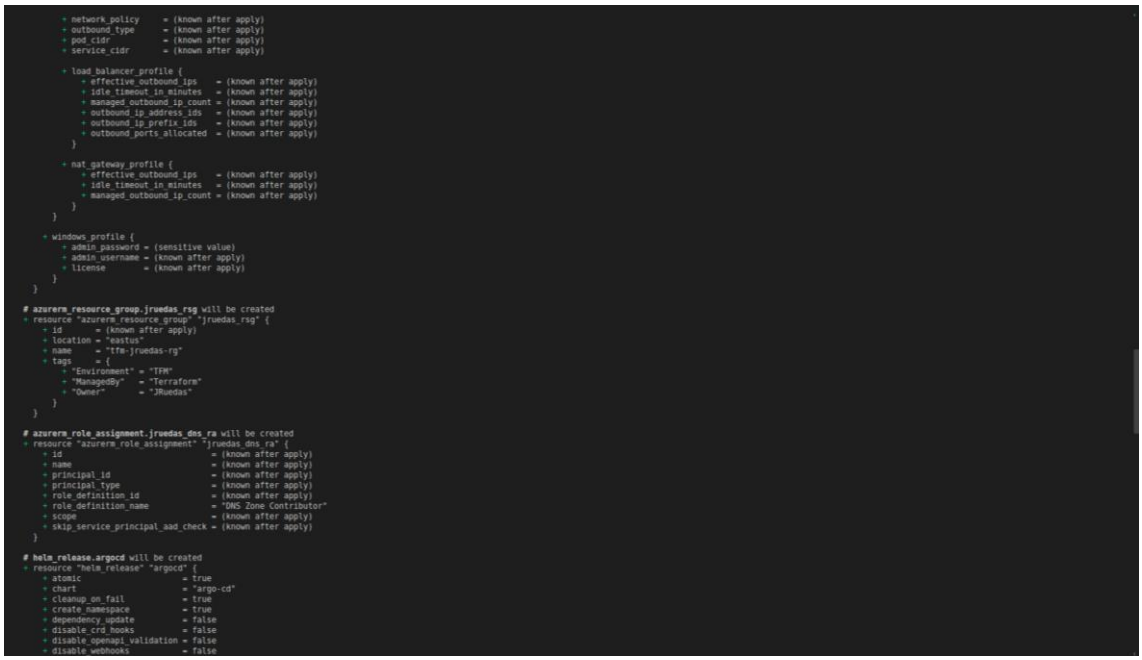




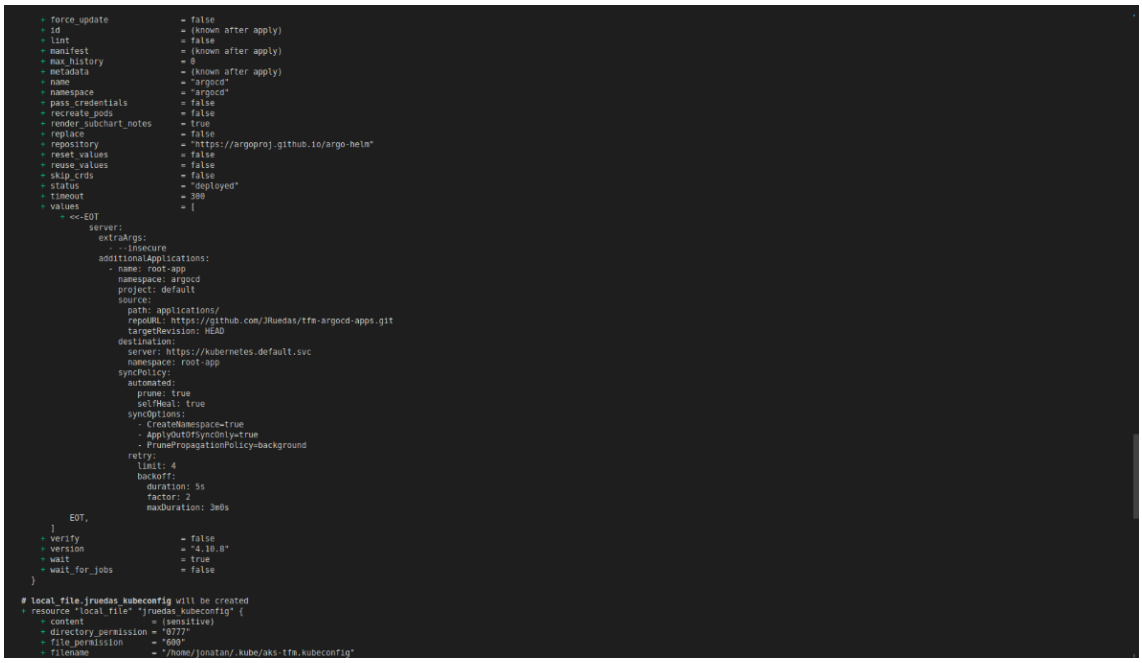
```
  }

# azurerm_key_vault_access_policy.jruedas_aks_kvp will be created
+ resource "azurerm_key_vault_access_policy" "jruedas_aks_kvp" {
    + id                  = (known after apply)
    + key_permissions     = [
        + "Get",
      ]
    + key_vault_id        = (known after apply)
    + object_id           = (known after apply)
    + secret_permissions = [
        + "Get",
      ]
    + tenant_id           = "[redacted]"
  }

# azurerm_key_vault_secret.cert_manager_secret will be created
+ resource "azurerm_key_vault_secret" "cert_manager_secret" {
    + id                  = (known after apply)
    + key_vault_id        = (known after apply)
    + name                = "cert-manager-secret"
    + resource_id         = (known after apply)
    + resource_versionless_id = (known after apply)
    + value               = (sensitive value)
    + version             = (known after apply)
    + versionless_id      = (known after apply)
  }

# azurerm_key_vault_secret.external_dns_secret will be created
+ resource "azurerm_key_vault_secret" "external_dns_secret" {
    + id                  = (known after apply)
    + key_vault_id        = (known after apply)
    + name                = "external-dns-secret"
    + resource_id         = (known after apply)
    + resource_versionless_id = (known after apply)
    + value               = (sensitive value)
    + version             = (known after apply)
    + versionless_id      = (known after apply)
  }

# azurerm_kubernetes_cluster.jruedas_aks will be created
+ resource "azurerm_kubernetes_cluster" "jruedas_aks" {
    + dns_prefix                        = "tmf-jruedas"
    + fqdn                              = (known after apply)
    + http_application_routing_zone_name  = (known after apply)
    + id                                = (known after apply)
    + kube_admin_config                 = (sensitive value)
    + kube_admin_config_raw             = (sensitive value)
    + kube_config                       = (sensitive value)
    + kube_config_raw                   = (sensitive value)
    + kubernetes_version                = "1.22"
    + location                          = "eastus"
    + name                              = "tfm-jruedas-aks"
    + node_resource_group               = (known after apply)
    + oidc_issuer_url                   = (known after apply)
    + portal_fqdn                       = (known after apply)
    + private_cluster_public_fqdn_enabled = false
    + private_dns_zone_id               = (known after apply)
    + private_fqdn                      = (known after apply)
    + public_network_access_enabled     = true
    + resource_group_name               = "tfm-jruedas-rg"
    + role_based_access_control_enabled   = true
```



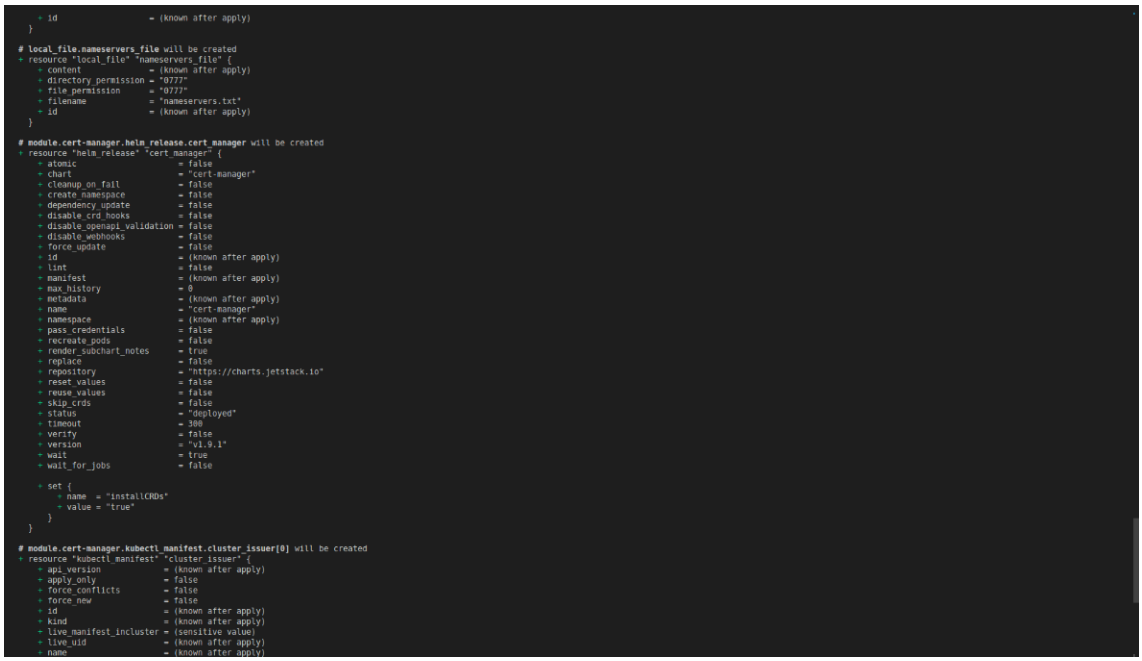
```
    + run_command_enabled               = true
    + sku_tier                          = "Free"
    + tags                              = {
        + "Environment" = "TFM"
        + "ManagedBy"   = "Terraform"
        + "Owner"       = "JRuedas"
      }

    + auto_scaler_profile {
        + balance_similar_node_groups      = (known after apply)
        + empty_bulk_delete_max            = (known after apply)
        + expander                         = (known after apply)
        + max_graceful_termination_sec     = (known after apply)
        + max_node_provisioning_time       = (known after apply)
        + max_unready_nodes                = (known after apply)
        + max_unready_percentage           = (known after apply)
        + new_pod_scale_up_delay           = (known after apply)
        + scale_down_delay_after_add       = (known after apply)
        + scale_down_delay_after_delete    = (known after apply)
        + scale_down_delay_after_failure   = (known after apply)
        + scale_down_unneeded              = (known after apply)
        + scale_down_unready               = (known after apply)
        + scale_down_utilization_threshold = (known after apply)
        + scan_interval                    = (known after apply)
        + skip_nodes_with_local_storage    = (known after apply)
        + skip_nodes_with_system_pods      = (known after apply)
      }

    + default_node_pool {
        + kubelet_disk_type    = (known after apply)
        + max_pods             = (known after apply)
        + name                 = "tfmjruedasnp"
        + node_count           = 2
        + node_labels          = (known after apply)
        + orchestrator_version = (known after apply)
        + os_disk_size_gb      = (known after apply)
        + os_disk_type         = "Managed"
        + os_sku               = (known after apply)
        + type                 = "VirtualMachineScaleSets"
        + ultra_ssd_enabled    = false
        + vm_size              = "Standard_B2s"
      }

    + identity {
        + principal_id = (known after apply)
        + tenant_id    = (known after apply)
        + type         = "SystemAssigned"
      }

    + kubelet_identity {
        + client_id                = (known after apply)
        + object_id                = (known after apply)
        + user_assigned_identity_id = (known after apply)
      }

    + network_profile {
        + dns_service_ip     = (known after apply)
        + docker_bridge_cidr = (known after apply)
        + ip_versions        = (known after apply)
        + load_balancer_sku  = (known after apply)
        + network_mode       = (known after apply)
        + network_plugin     = (known after apply)
```

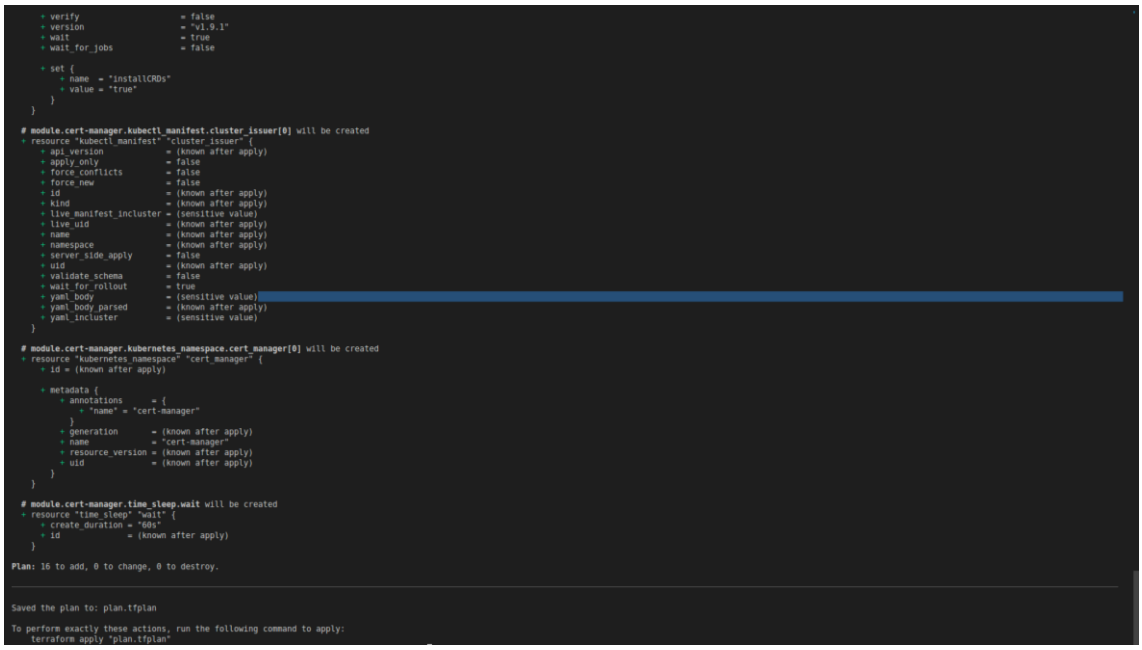


```
      + network_policy     = (known after apply)
      + outbound_type      = (known after apply)
      + pod_cidr           = (known after apply)
      + service_cidr       = (known after apply)

      + load_balancer_profile {
          + effective_outbound_ips    = (known after apply)
          + idle_timeout_in_minutes   = (known after apply)
          + managed_outbound_ip_count = (known after apply)
          + outbound_ip_address_ids   = (known after apply)
          + outbound_ip_prefix_ids    = (known after apply)
          + outbound_ports_allocated  = (known after apply)
        }

      + nat_gateway_profile {
          + effective_outbound_ips    = (known after apply)
          + idle_timeout_in_minutes   = (known after apply)
          + managed_outbound_ip_count = (known after apply)
        }
    }

  + windows_profile {
      + admin_password = (sensitive value)
      + admin_username = (known after apply)
      + license        = (known after apply)
    }
}

# azurerm_resource_group.jruedas_rsg will be created
+ resource "azurerm_resource_group" "jruedas_rsg" {
    + id       = (known after apply)
    + location = "eastus"
    + name     = "tfm-jruedas-rg"
    + tags     = {
        + "Environment" = "TFM"
        + "ManagedBy"   = "Terraform"
        + "Owner"       = "JRuedas"
      }
  }

# azurerm_role_assignment.jruedas_dns_ra will be created
+ resource "azurerm_role_assignment" "jruedas_dns_ra" {
    + id                               = (known after apply)
    + name                             = (known after apply)
    + principal_id                     = (known after apply)
    + principal_type                   = (known after apply)
    + role_definition_id               = (known after apply)
    + role_definition_name             = "DNS Zone Contributor"
    + scope                            = (known after apply)
    + skip_service_principal_aad_check = (known after apply)
  }

# helm_release.argocd will be created
+ resource "helm_release" "argocd" {
    + atomic                     = true
    + chart                      = "argo-cd"
    + cleanup_on_fail            = true
    + create_namespace           = true
    + dependency_update          = false
    + disable_crd_hooks          = false
    + disable_openapi_validation = false
    + disable_webhooks           = false
```



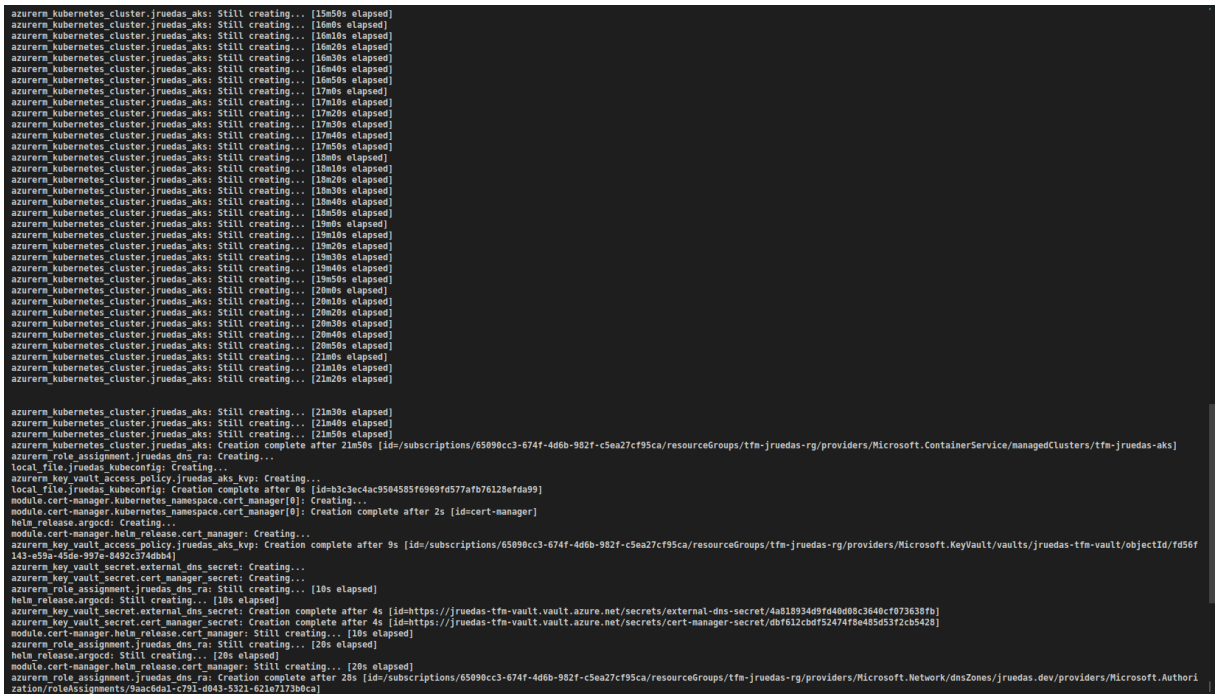
```
    + force_update               = false
    + id                         = (known after apply)
    + lint                       = false
    + manifest                   = (known after apply)
    + max_history                = 0
    + metadata                   = (known after apply)
    + name                       = "argocd"
    + namespace                  = "argocd"
    + pass_credentials           = false
    + recreate_pods              = false
    + render_subchart_notes      = true
    + replace                    = false
    + repository                 = "https://argoproj.github.io/argo-helm"
    + reset_values               = false
    + reuse_values               = false
    + skip_crds                  = false
    + status                     = "deployed"
    + timeout                    = 300
    + values                     = [
        + <<-EOT
            server:
              extraArgs:
                - --insecure
              additionalApplications:
                - name: root-app
                  namespace: argocd
                  project: default
                  source:
                    path: applications/
                    repoURL: https://github.com/JRuedas/tfm-argocd-apps.git
                    targetRevision: HEAD
                  destination:
                    server: https://kubernetes.default.svc
                    namespace: root-app
                  syncPolicy:
                    automated:
                      prune: true
                      selfHeal: true
                    syncOptions:
                      - CreateNamespace=true
                      - ApplyOutOfSyncOnly=true
                      - PrunePropagationPolicy=background
                    retry:
                      limit: 4
                      backoff:
                        duration: 5s
                        factor: 2
                        maxDuration: 3m0s
            EOT,
      ]
    + verify                     = false
    + version                    = "4.10.8"
    + wait                       = true
    + wait_for_jobs              = false
  }

# local_file.jruedas_kubeconfig will be created
+ resource "local_file" "jruedas_kubeconfig" {
    + content              = (sensitive)
    + directory_permission = "0777"
    + file_permission      = "600"
    + filename             = "/home/jonatan/.kube/aks-tfm.kubeconfig"
```

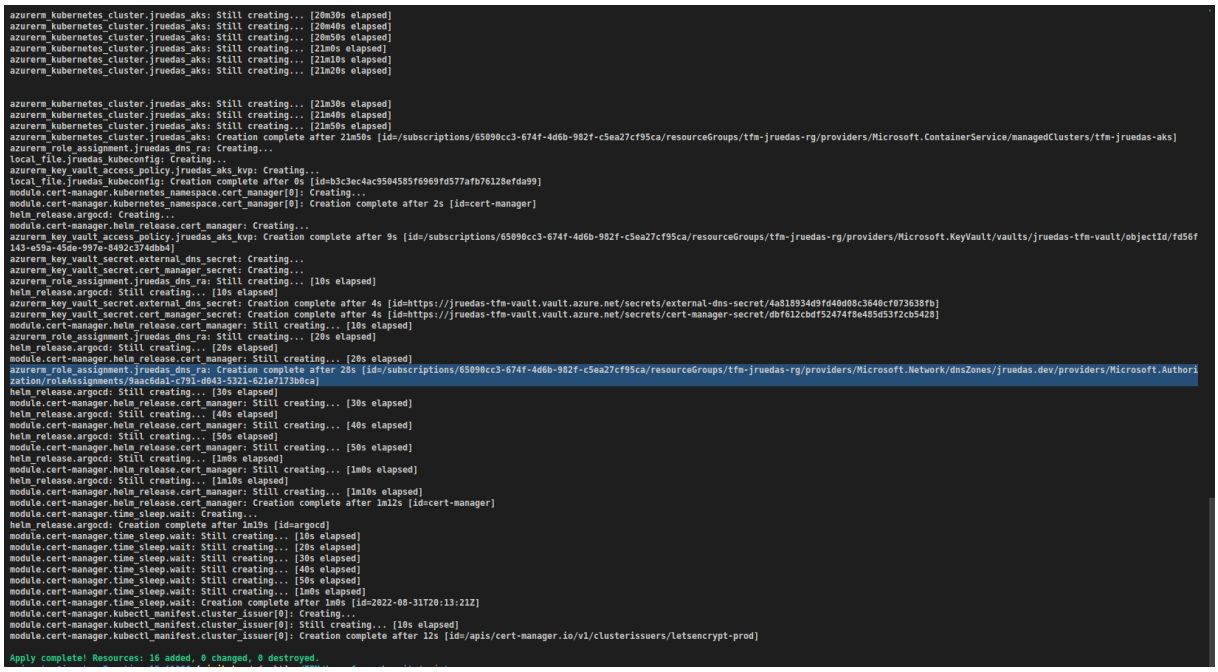

+ id = (known after apply)
}
# local_file.nameservers_file will be created
+ resource "local_file" "nameservers_file" {
+ content = (known after apply)
+ directory_permission = "0777"
+ file_permission = "0777"
+ filename = "nameservers.txt"
+ id = (known after apply)
}
# module.cert-manager.helm_release.cert_manager will be created
+ resource "helm_release" "cert_manager" {
+ atomic = false
+ chart = "cert-manager"
+ cleanup_on_fail = false
+ create_namespace = false
+ dependency_update = false
+ disable_crd_hooks = false
+ disable_openapi_validation = false
+ disable_webhooks = false
+ force_update = false
+ id = (known after apply)
+ lint = false
+ manifest = (known after apply)
+ max_history = 0
+ metadata = (known after apply)
+ name = "cert-manager"
+ namespace = (known after apply)
+ pass_credentials = false
+ recreate_pods = false
+ render_subchart_notes = true
+ replace = false
+ repository = "https://charts.jetstack.io"
+ reset_values = false
+ reuse_values = false
+ skip_crds = false
+ status = "deployed"
+ timeout = 300
+ verify = false
+ version = "v1.9.1"
+ wait = true
+ wait_for_jobs = false
+ set {
+ name = "installCRDs"
+ value = "true"
}
}
# module.cert-manager.kubectl_manifest.cluster_issuer[0] will be created
+ resource "kubectl_manifest" "cluster_issuer" {
+ api_version = (known after apply)
+ apply_only = false
+ force_conflicts = false
+ force_new = false
+ id = (known after apply)
+ kind = (known after apply)
+ live_manifest_incluster = (sensitive value)
+ live_uid = (known after apply)
+ name = (known after apply)


+ verify = false
+ version = "v1.9.1"
+ wait = true
+ wait_for_jobs = false
+ set {
+ name = "installCRDs"
+ value = "true"
}
}
# module.cert-manager.kubectl_manifest.cluster_issuer[0] will be created
+ resource "kubectl_manifest" "cluster_issuer" {
+ api_version = (known after apply)
+ apply_only = false
+ force_conflicts = false
+ force_new = false
+ id = (known after apply)
+ kind = (known after apply)
+ live_manifest_incluster = (sensitive value)
+ live_uid = (known after apply)
+ name = (known after apply)
+ namespace = (known after apply)
+ server_side_apply = false
+ uid = (known after apply)
+ validate_schema = false
+ wait_for_rollout = true
+ yaml_body = (sensitive value)
+ yaml_body_parsed = (known after apply)
+ yaml_incluster = (sensitive value)
}
# module.cert-manager.kubernetes_namespace.cert_manager[0] will be created
+ resource "kubernetes_namespace" "cert_manager" {
+ id = (known after apply)
+ metadata {
+ annotations = {
+ "name" = "cert-manager"
}
+ generation = (known after apply)
+ name = "cert-manager"
+ resource_version = (known after apply)
+ uid = (known after apply)
}
}
# module.cert-manager.time_sleep.wait will be created
+ resource "time_sleep" "wait" {
+ create_duration = "60s"
+ id = (known after apply)
}
Plan: 16 to add, 0 to change, 0 to destroy.
Saved the plan to: plan.tfplan
To perform exactly these actions, run the following command to apply:
terraform apply "plan.tfplan"

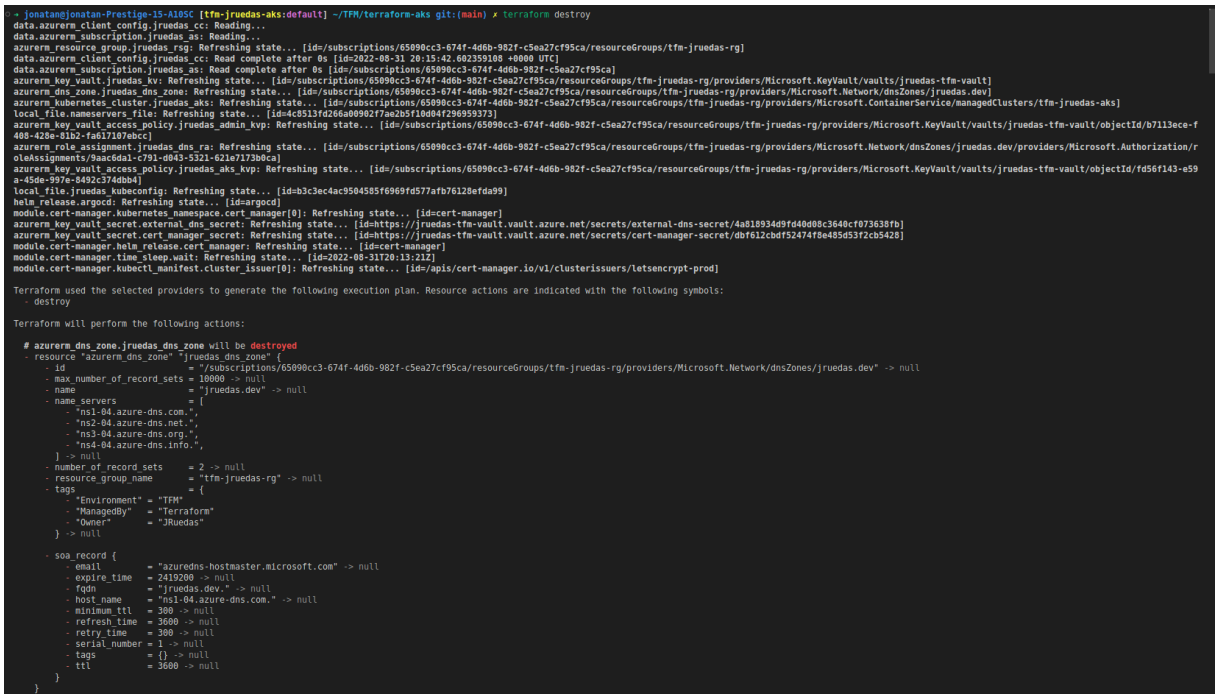
A-5. Aplicación del plan

```
➜ jonatangjonatan-Prestige-15-A10SC [minikube:default] ~/TFM/terraform-aks git:(main) terraform apply "plan.tfplan"
azurerm_resource_group.jruedas_rsg: Creating...
azurerm_resource_group.jruedas_rsg: Creation complete after 3s [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg]
azurerm_dns_zone.jruedas_dns_zone: Creating...
azurerm_key_vault.jruedas_kv: Creating...
azurerm_kubernetes_cluster.jruedas_aks: Creating...
azurerm_dns_zone.jruedas_dns_zone: Creation complete after 7s [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.Network/dnsZones/jruedas.dev]
local_file.nameservers_file: Creating...
local_file.nameservers_file: Creation complete after 0s [id=4c8513fd266a00902f7ae2b5f10d04f296959373]
azurerm_key_vault.jruedas_kv: Still creating... [10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [10s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [20s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [30s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [40s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [50s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [1m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [1m0s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [1m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [1m10s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [1m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [1m20s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [1m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [1m30s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [1m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [1m40s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [1m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [1m50s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [2m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [2m0s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [2m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [2m10s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [2m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [2m20s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [2m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [2m30s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [2m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [2m40s elapsed]
azurerm_key_vault.jruedas_kv: Still creating... [2m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [2m50s elapsed]
azurerm_key_vault.jruedas_kv: Creation complete after 2m52s [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault]
azurerm_key_vault_access_policy.jruedas_admin_kvp: Creating...
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [3m0s elapsed]
azurerm_key_vault_access_policy.jruedas_admin_kvp: Creation complete after 8s [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault/objectId/b713ece-f408-428e-81b2-fa617107ebcc]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [3m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [3m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [3m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [3m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [3m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [4m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [4m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [4m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [4m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [4m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [4m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [5m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [5m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [5m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [5m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [5m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [5m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [6m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [6m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [6m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [6m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [6m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [6m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [7m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [7m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [7m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [7m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [7m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [7m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [8m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [8m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [8m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [8m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [8m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [8m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [9m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [9m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [9m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [9m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [9m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [9m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [10m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [10m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [10m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [10m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [10m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [10m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [11m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [11m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [11m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [11m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [11m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [11m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [12m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [12m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [12m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [12m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [12m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [12m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [13m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [13m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [13m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [13m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [13m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [13m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [14m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [14m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [14m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [14m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [14m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [14m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [15m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [15m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [15m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [15m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still creating... [15m40s elapsed]
```

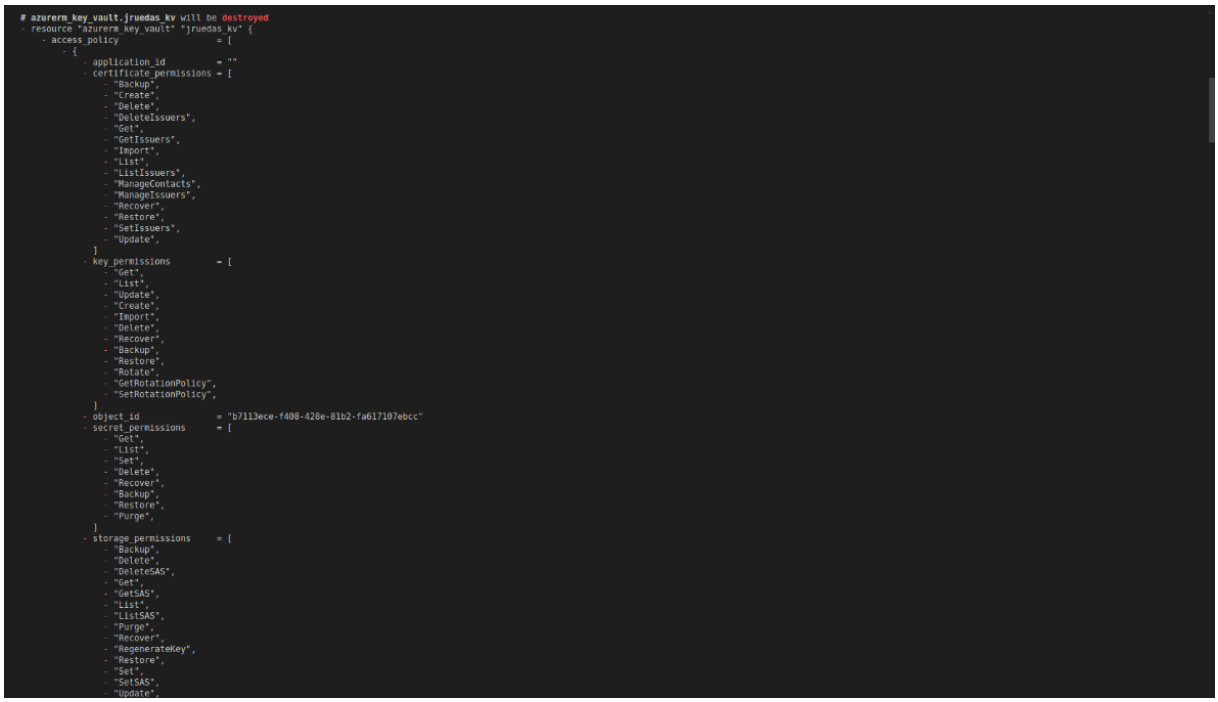




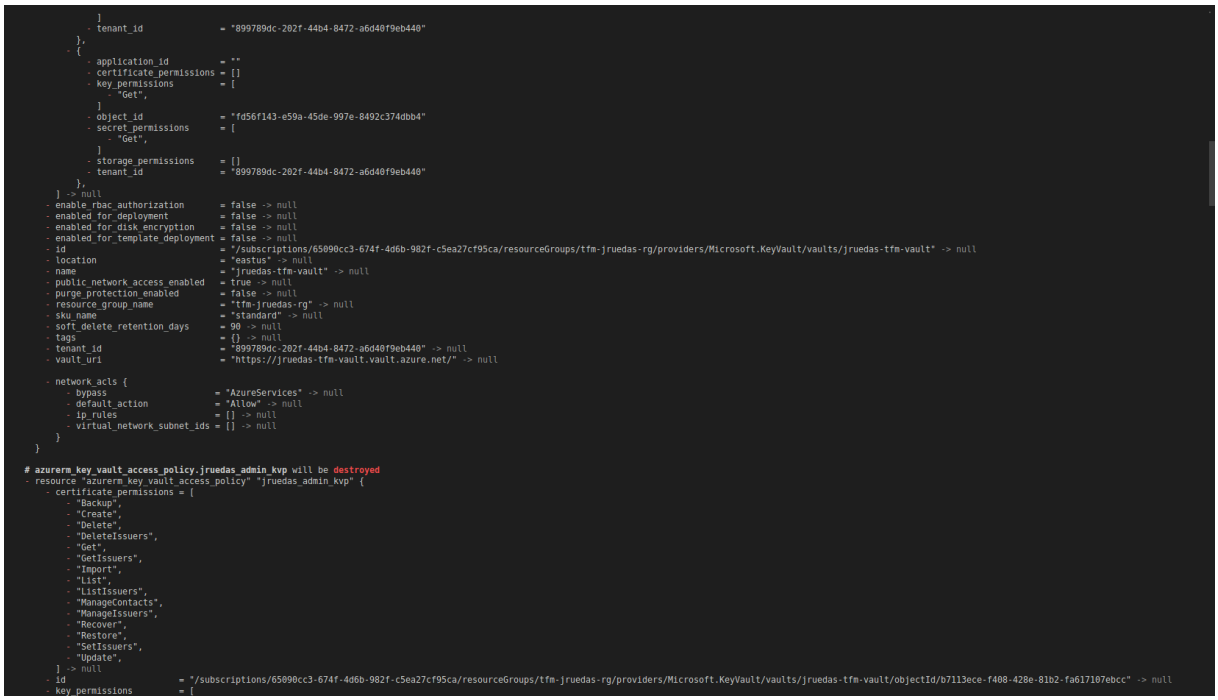
A-6. Destrucción de los recursos



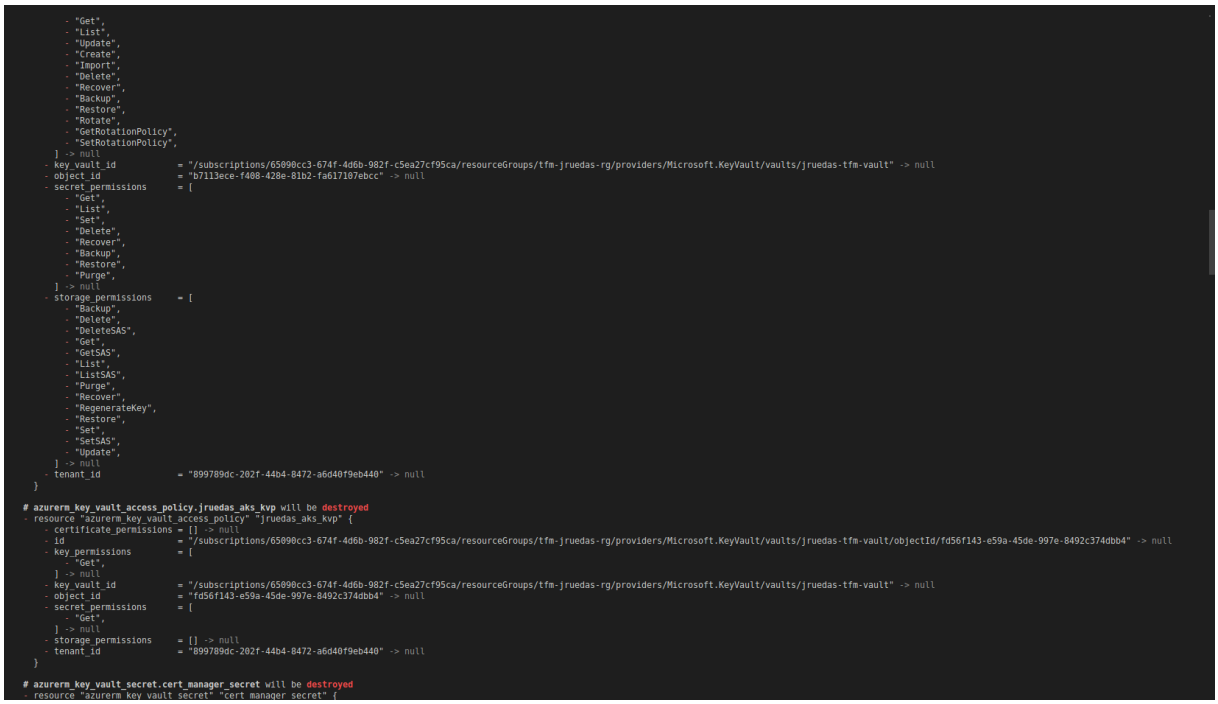
```
⚙ → jonatan@jonatan-Prestige-15-A10SC [tfm-jruedas-aks:default] ~/TFM/terraform-aks git:(main) ✗ terraform destroy
data.azurerm_client_config.jruedas_cc: Reading...
data.azurerm_subscription.jruedas_as: Reading...
azurerm_resource_group.jruedas_rsg: Refreshing state... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg]
data.azurerm_client_config.jruedas_cc: Read complete after 0s [id=2022-08-31 20:15:42.602359108 +0000 UTC]
data.azurerm_subscription.jruedas_as: Read complete after 0s [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca]
azurerm_key_vault.jruedas_kv: Refreshing state... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault]
azurerm_dns_zone.jruedas_dns_zone: Refreshing state... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.Network/dnsZones/jruedas.dev]
azurerm_kubernetes_cluster.jruedas_aks: Refreshing state... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.ContainerService/managedClusters/tfm-jruedas-aks]
local_file.nameservers_file: Refreshing state... [id=4c8513fd266a00902f7ae2b5f10d04f296959373]
azurerm_key_vault_access_policy.jruedas_admin_kvp: Refreshing state... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault/objectId/b7113ece-f408-428e-81b2-fa617107ebcc]
azurerm_role_assignment.jruedas_dns_ra: Refreshing state... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.Network/dnsZones/jruedas.dev/providers/Microsoft.Authorization/roleAssignments/9aac6da1-c791-d043-5321-621e7173b0ca]
azurerm_key_vault_access_policy.jruedas_aks_kvp: Refreshing state... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault/objectId/fd56f143-e59a-45de-997e-8492c374dbb4]
local_file.jruedas_kubeconfig: Refreshing state... [id=b3c3ec4ac9504585f6969fd577afb76128efda99]
helm_release.argocd: Refreshing state... [id=argocd]
module.cert-manager.kubernetes_namespace.cert_manager[0]: Refreshing state... [id=cert-manager]
azurerm_key_vault_secret.external_dns_secret: Refreshing state... [id=https://jruedas-tfm-vault.vault.azure.net/secrets/external-dns-secret/4a818934d9fd40d08c3640cf073638fb]
azurerm_key_vault_secret.cert_manager_secret: Refreshing state... [id=https://jruedas-tfm-vault.vault.azure.net/secrets/cert-manager-secret/dbf612cbdf52474f8e485d53f2cb5428]
module.cert-manager.helm_release.cert_manager: Refreshing state... [id=cert-manager]
module.cert-manager.time_sleep.wait: Refreshing state... [id=2022-08-31T20:13:21Z]
module.cert-manager.kubectl_manifest.cluster_issuer[0]: Refreshing state... [id=/apis/cert-manager.io/v1/clusterissuers/letsencrypt-prod]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # azurerm_dns_zone.jruedas_dns_zone will be destroyed
  - resource "azurerm_dns_zone" "jruedas_dns_zone" {
      - id                      = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.Network/dnsZones/jruedas.dev" -> null
      - max_number_of_record_sets = 10000 -> null
      - name                    = "jruedas.dev" -> null
      - name_servers            = [
          - "ns1-04.azure-dns.com.",
          - "ns2-04.azure-dns.net.",
          - "ns3-04.azure-dns.org.",
          - "ns4-04.azure-dns.info.",
        ] -> null
      - number_of_record_sets     = 2 -> null
      - resource_group_name       = "tfm-jruedas-rg" -> null
      - tags                    = {
          - "Environment" = "TFM"
          - "ManagedBy"   = "Terraform"
          - "Owner"       = "JRuedas"
        } -> null

      - soa_record {
          - email         = "azuredns-hostmaster.microsoft.com" -> null
          - expire_time   = 2419200 -> null
          - fqdn          = "jruedas.dev." -> null
          - host_name     = "ns1-04.azure-dns.com." -> null
          - minimum_ttl   = 300 -> null
          - refresh_time  = 3600 -> null
          - retry_time    = 300 -> null
          - serial_number = 1 -> null
          - tags          = {} -> null
          - ttl           = 3600 -> null
        }
    }
```



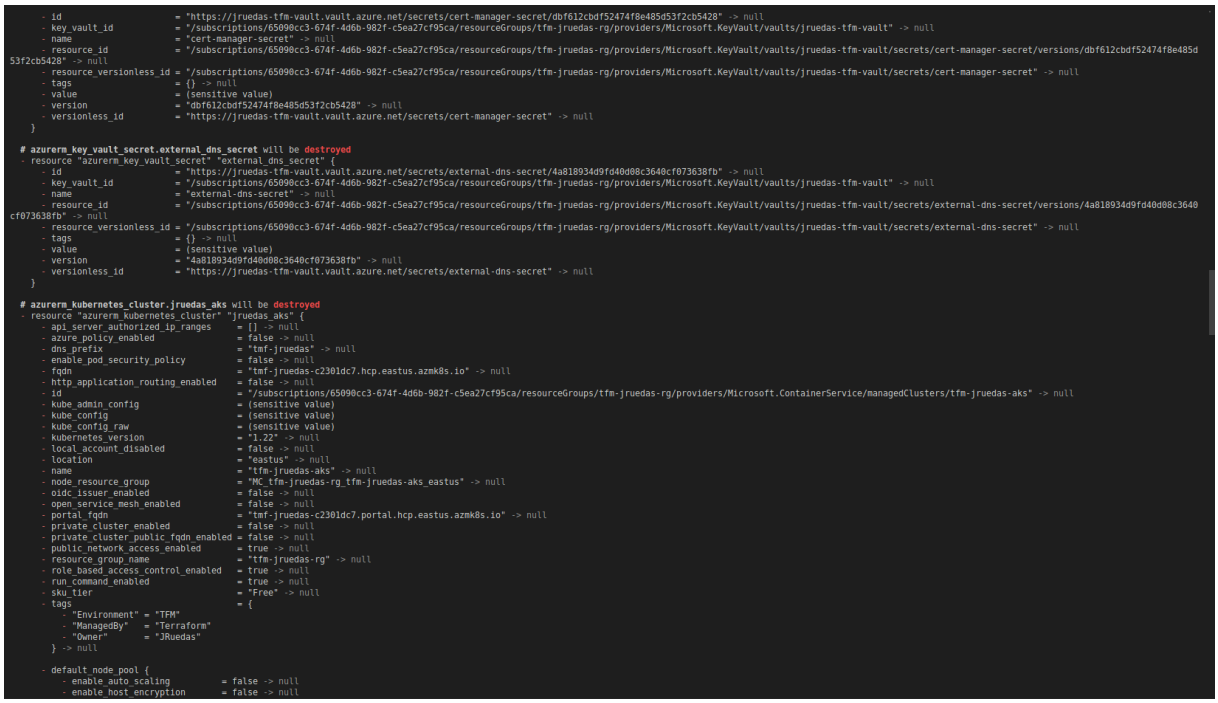
```
  # azurerm_key_vault.jruedas_kv will be destroyed
  - resource "azurerm_key_vault" "jruedas_kv" {
      - access_policy             = [
          - {
              - application_id          = ""
              - certificate_permissions = [
                  - "Backup",
                  - "Create",
                  - "Delete",
                  - "DeleteIssuers",
                  - "Get",
                  - "GetIssuers",
                  - "Import",
                  - "List",
                  - "ListIssuers",
                  - "ManageContacts",
                  - "ManageIssuers",
                  - "Recover",
                  - "Restore",
                  - "SetIssuers",
                  - "Update",
                ]
              - key_permissions         = [
                  - "Get",
                  - "List",
                  - "Update",
                  - "Create",
                  - "Import",
                  - "Delete",
                  - "Recover",
                  - "Backup",
                  - "Restore",
                  - "Rotate",
                  - "GetRotationPolicy",
                  - "SetRotationPolicy",
                ]
              - object_id               = "b7113ece-f408-428e-81b2-fa617107ebcc"
              - secret_permissions      = [
                  - "Get",
                  - "List",
                  - "Set",
                  - "Delete",
                  - "Recover",
                  - "Backup",
                  - "Restore",
                  - "Purge",
                ]
              - storage_permissions     = [
                  - "Backup",
                  - "Delete",
                  - "DeleteSAS",
                  - "Get",
                  - "GetSAS",
                  - "List",
                  - "ListSAS",
                  - "Purge",
                  - "Recover",
                  - "RegenerateKey",
                  - "Restore",
                  - "Set",
                  - "SetSAS",
                  - "Update",
```



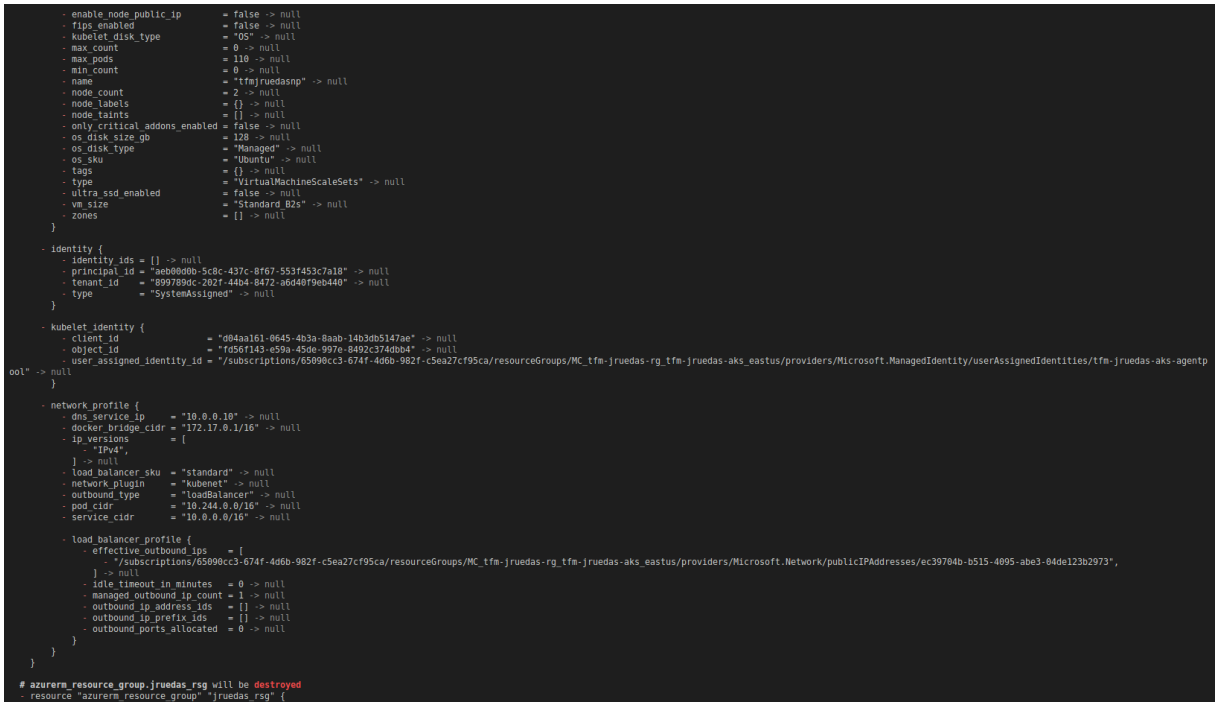
```
            ]
          - tenant_id               = "899789dc-202f-44b4-8472-a6d40f9eb440"
        },
      - {
          - application_id          = ""
          - certificate_permissions = []
          - key_permissions         = [
              - "Get",
            ]
          - object_id               = "fd56f143-e59a-45de-997e-8492c374dbb4"
          - secret_permissions      = [
              - "Get",
            ]
          - storage_permissions     = []
          - tenant_id               = "899789dc-202f-44b4-8472-a6d40f9eb440"
        },
    ] -> null
    - enable_rbac_authorization       = false -> null
    - enabled_for_deployment          = false -> null
    - enabled_for_disk_encryption     = false -> null
    - enabled_for_template_deployment = false -> null
    - id                              = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault" -> null
    - location                        = "eastus" -> null
    - name                            = "jruedas-tfm-vault" -> null
    - public_network_access_enabled   = true -> null
    - purge_protection_enabled        = false -> null
    - resource_group_name             = "tfm-jruedas-rg" -> null
    - sku_name                        = "standard" -> null
    - soft_delete_retention_days      = 90 -> null
    - tags                            = {} -> null
    - tenant_id                       = "899789dc-202f-44b4-8472-a6d40f9eb440" -> null
    - vault_uri                       = "https://jruedas-tfm-vault.vault.azure.net/" -> null

    - network_acls {
        - bypass                  = "AzureServices" -> null
        - default_action          = "Allow" -> null
        - ip_rules                = [] -> null
        - virtual_network_subnet_ids = [] -> null
      }
  }

# azurerm_key_vault_access_policy.jruedas_admin_kvp will be destroyed
- resource "azurerm_key_vault_access_policy" "jruedas_admin_kvp" {
    - certificate_permissions = [
        - "Backup",
        - "Create",
        - "Delete",
        - "DeleteIssuers",
        - "Get",
        - "GetIssuers",
        - "Import",
        - "List",
        - "ListIssuers",
        - "ManageContacts",
        - "ManageIssuers",
        - "Recover",
        - "Restore",
        - "SetIssuers",
        - "Update",
      ] -> null
    - id                      = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault/objectId/b7113ece-f408-428e-81b2-fa617107ebcc" -> null
    - key_permissions         = [
```



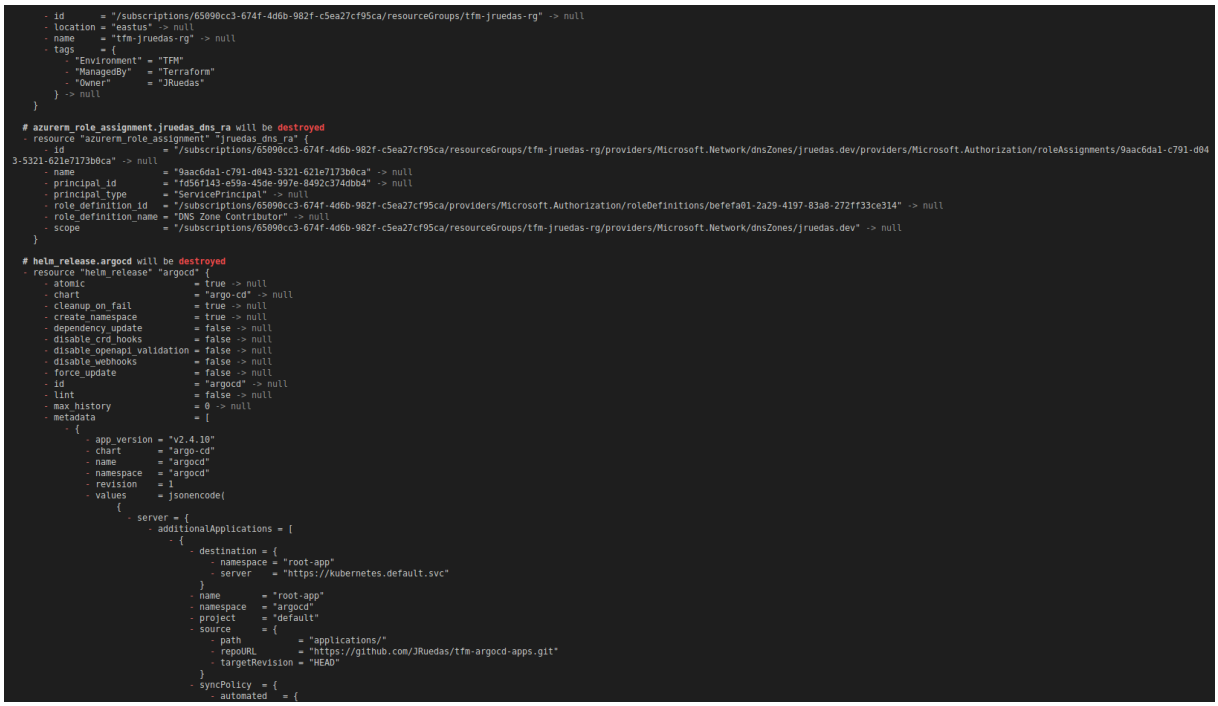
```
        - "Get",
        - "List",
        - "Update",
        - "Create",
        - "Import",
        - "Delete",
        - "Recover",
        - "Backup",
        - "Restore",
        - "Rotate",
        - "GetRotationPolicy",
        - "SetRotationPolicy",
      ] -> null
    - key_vault_id            = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault" -> null
    - object_id               = "b7113ece-f408-428e-81b2-fa617107ebcc" -> null
    - secret_permissions      = [
        - "Get",
        - "List",
        - "Set",
        - "Delete",
        - "Recover",
        - "Backup",
        - "Restore",
        - "Purge",
      ] -> null
    - storage_permissions     = [
        - "Backup",
        - "Delete",
        - "DeleteSAS",
        - "Get",
        - "GetSAS",
        - "List",
        - "ListSAS",
        - "Purge",
        - "Recover",
        - "RegenerateKey",
        - "Restore",
        - "Set",
        - "SetSAS",
        - "Update",
      ] -> null
    - tenant_id               = "899789dc-202f-44b4-8472-a6d40f9eb440" -> null
  }

# azurerm_key_vault_access_policy.jruedas_aks_kvp will be destroyed
- resource "azurerm_key_vault_access_policy" "jruedas_aks_kvp" {
    - certificate_permissions = [] -> null
    - id                      = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault/objectId/fd56f143-e59a-45de-997e-8492c374dbb4" -> null
    - key_permissions         = [
        - "Get",
      ] -> null
    - key_vault_id            = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault" -> null
    - object_id               = "fd56f143-e59a-45de-997e-8492c374dbb4" -> null
    - secret_permissions      = [
        - "Get",
      ] -> null
    - storage_permissions     = [] -> null
    - tenant_id               = "899789dc-202f-44b4-8472-a6d40f9eb440" -> null
  }

# azurerm_key_vault_secret.cert_manager_secret will be destroyed
- resource "azurerm_key_vault_secret" "cert_manager_secret" {
```



```
      - id                    = "https://jruedas-tfm-vault.vault.azure.net/secrets/cert-manager-secret/dbf612cbdf52474f8e485d53f2cb5428" -> null
      - key_vault_id          = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault" -> null
      - name                  = "cert-manager-secret" -> null
      - resource_id           = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault/secrets/cert-manager-secret/versions/dbf612cbdf52474f8e485d
53f2cb5428" -> null
      - resource_versionless_id = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault/secrets/cert-manager-secret" -> null
      - tags                  = {} -> null
      - value                 = (sensitive value)
      - version               = "dbf612cbdf52474f8e485d53f2cb5428" -> null
      - versionless_id        = "https://jruedas-tfm-vault.vault.azure.net/secrets/cert-manager-secret" -> null
    }

  # azurerm_key_vault_secret.external_dns_secret will be destroyed
  - resource "azurerm_key_vault_secret" "external_dns_secret" {
      - id                    = "https://jruedas-tfm-vault.vault.azure.net/secrets/external-dns-secret/4a818934d9fd40d08c3640cf073638fb" -> null
      - key_vault_id          = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault" -> null
      - name                  = "external-dns-secret" -> null
      - resource_id           = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault/secrets/external-dns-secret/versions/4a818934d9fd40d08c3640
cf073638fb" -> null
      - resource_versionless_id = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault/secrets/external-dns-secret" -> null
      - tags                  = {} -> null
      - value                 = (sensitive value)
      - version               = "4a818934d9fd40d08c3640cf073638fb" -> null
      - versionless_id        = "https://jruedas-tfm-vault.vault.azure.net/secrets/external-dns-secret" -> null
    }

  # azurerm_kubernetes_cluster.jruedas_aks will be destroyed
  - resource "azurerm_kubernetes_cluster" "jruedas_aks" {
      - api_server_authorized_ip_ranges     = [] -> null
      - azure_policy_enabled                = false -> null
      - dns_prefix                          = "tmf-jruedas" -> null
      - enable_pod_security_policy          = false -> null
      - fqdn                                = "tmf-jruedas-c2301dc7.hcp.eastus.azmk8s.io" -> null
      - http_application_routing_enabled    = false -> null
      - id                                  = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.ContainerService/managedClusters/tfm-jruedas-aks" -> null
      - kube_admin_config                   = (sensitive value)
      - kube_config                         = (sensitive value)
      - kube_config_raw                     = (sensitive value)
      - kubernetes_version                  = "1.22" -> null
      - local_account_disabled              = false -> null
      - location                            = "eastus" -> null
      - name                                = "tfm-jruedas-aks" -> null
      - node_resource_group                 = "MC_tfm-jruedas-rg_tfm-jruedas-aks_eastus" -> null
      - oidc_issuer_enabled                 = false -> null
      - open_service_mesh_enabled           = false -> null
      - portal_fqdn                         = "tmf-jruedas-c2301dc7.portal.hcp.eastus.azmk8s.io" -> null
      - private_cluster_enabled             = false -> null
      - private_cluster_public_fqdn_enabled = false -> null
      - public_network_access_enabled       = true -> null
      - resource_group_name                 = "tfm-jruedas-rg" -> null
      - role_based_access_control_enabled   = true -> null
      - run_command_enabled                 = true -> null
      - sku_tier                            = "Free" -> null
      - tags                                = {
          - "Environment" = "TFM"
          - "ManagedBy"   = "Terraform"
          - "Owner"       = "JRuedas"
        } -> null

      - default_node_pool {
          - enable_auto_scaling          = false -> null
          - enable_host_encryption       = false -> null
```

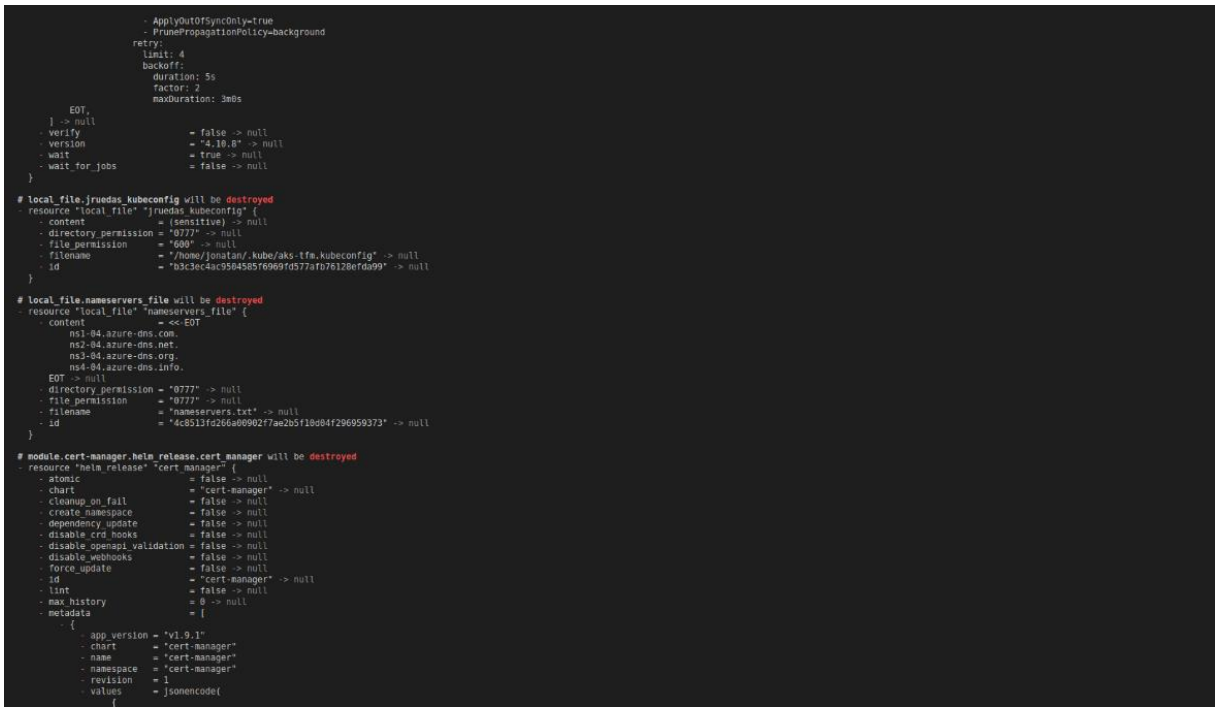


```
          - enable_node_public_ip        = false -> null
          - fips_enabled                 = false -> null
          - kubelet_disk_type            = "OS" -> null
          - max_count                    = 0 -> null
          - max_pods                     = 110 -> null
          - min_count                    = 0 -> null
          - name                         = "tfmjruedasnp" -> null
          - node_count                   = 2 -> null
          - node_labels                  = {} -> null
          - node_taints                  = [] -> null
          - only_critical_addons_enabled = false -> null
          - os_disk_size_gb              = 128 -> null
          - os_disk_type                 = "Managed" -> null
          - os_sku                       = "Ubuntu" -> null
          - tags                         = {} -> null
          - type                         = "VirtualMachineScaleSets" -> null
          - ultra_ssd_enabled            = false -> null
          - vm_size                      = "Standard_B2s" -> null
          - zones                        = [] -> null
        }

      - identity {
          - identity_ids = [] -> null
          - principal_id = "aeb00d0b-5c8c-437c-8f67-553f453c7a18" -> null
          - tenant_id    = "899789dc-202f-44b4-8472-a6d40f9eb440" -> null
          - type         = "SystemAssigned" -> null
        }

      - kubelet_identity {
          - client_id                 = "d04aa161-0645-4b3a-8aab-14b3db5147ae" -> null
          - object_id                 = "fd56f143-e59a-45de-997e-8492c374dbb4" -> null
          - user_assigned_identity_id = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/MC_tfm-jruedas-rg_tfm-jruedas-aks_eastus/providers/Microsoft.ManagedIdentity/userAssignedIdentities/tfm-jruedas-aks-agentp
ool" -> null
        }

      - network_profile {
          - dns_service_ip     = "10.0.0.10" -> null
          - docker_bridge_cidr = "172.17.0.1/16" -> null
          - ip_versions        = [
              - "IPv4",
            ] -> null
          - load_balancer_sku  = "standard" -> null
          - network_plugin     = "kubenet" -> null
          - outbound_type      = "loadBalancer" -> null
          - pod_cidr           = "10.244.0.0/16" -> null
          - service_cidr       = "10.0.0.0/16" -> null

          - load_balancer_profile {
              - effective_outbound_ips    = [
                  - "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/MC_tfm-jruedas-rg_tfm-jruedas-aks_eastus/providers/Microsoft.Network/publicIPAddresses/ec39704b-b515-4095-abe3-04de123b2973",
                ] -> null
              - idle_timeout_in_minutes   = 0 -> null
              - managed_outbound_ip_count = 1 -> null
              - outbound_ip_address_ids   = [] -> null
              - outbound_ip_prefix_ids    = [] -> null
              - outbound_ports_allocated  = 0 -> null
            }
        }
    }

  # azurerm_resource_group.jruedas_rsg will be destroyed
  - resource "azurerm_resource_group" "jruedas_rsg" {
```

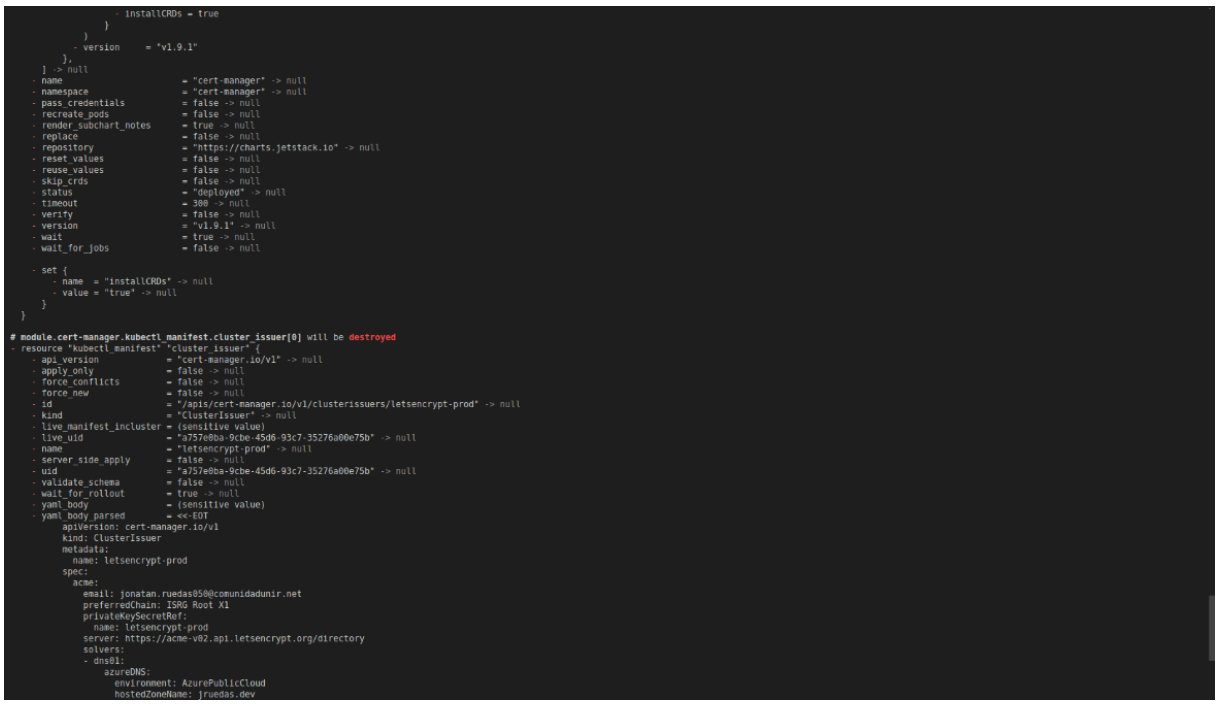


```
      - id       = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg" -> null
      - location = "eastus" -> null
      - name     = "tfm-jruedas-rg" -> null
      - tags     = {
          - "Environment" = "TFM"
          - "ManagedBy"   = "Terraform"
          - "Owner"       = "JRuedas"
        } -> null
    }

  # azurerm_role_assignment.jruedas_dns_ra will be destroyed
  - resource "azurerm_role_assignment" "jruedas_dns_ra" {
      - id                               = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.Network/dnsZones/jruedas.dev/providers/Microsoft.Authorization/roleAssignments/9aac6da1-c791-d043-5321-621e7173b0ca" -> null
      - name                             = "9aac6da1-c791-d043-5321-621e7173b0ca" -> null
      - principal_id                     = "fd56f143-e59a-45de-997e-8492c374dbb4" -> null
      - principal_type                   = "ServicePrincipal" -> null
      - role_definition_id               = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/providers/Microsoft.Authorization/roleDefinitions/befefa01-2a29-4197-83a8-272ff33ce314" -> null
      - role_definition_name             = "DNS Zone Contributor" -> null
      - scope                            = "/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.Network/dnsZones/jruedas.dev" -> null
    }

  # helm_release.argocd will be destroyed
  - resource "helm_release" "argocd" {
      - atomic                     = true -> null
      - chart                      = "argo-cd" -> null
      - cleanup_on_fail            = true -> null
      - create_namespace           = true -> null
      - dependency_update          = false -> null
      - disable_crd_hooks          = false -> null
      - disable_openapi_validation = false -> null
      - disable_webhooks           = false -> null
      - force_update               = false -> null
      - id                         = "argocd" -> null
      - lint                       = false -> null
      - max_history                = 0 -> null
      - metadata                   = [
          - {
              - app_version = "v2.4.10"
              - chart       = "argo-cd"
              - name        = "argocd"
              - namespace   = "argocd"
              - revision    = 1
              - values      = jsonencode(
                    {
                      - server = {
                          - additionalApplications = [
                              - {
                                  - destination = {
                                      - namespace = "root-app"
                                      - server    = "https://kubernetes.default.svc"
                                    }
                                  - name        = "root-app"
                                  - namespace   = "argocd"
                                  - project     = "default"
                                  - source      = {
                                      - path           = "applications/"
                                      - repoURL        = "https://github.com/JRuedas/tfm-argocd-apps.git"
                                      - targetRevision = "HEAD"
                                    }
                                  - syncPolicy  = {
                                      - automated   = {
```

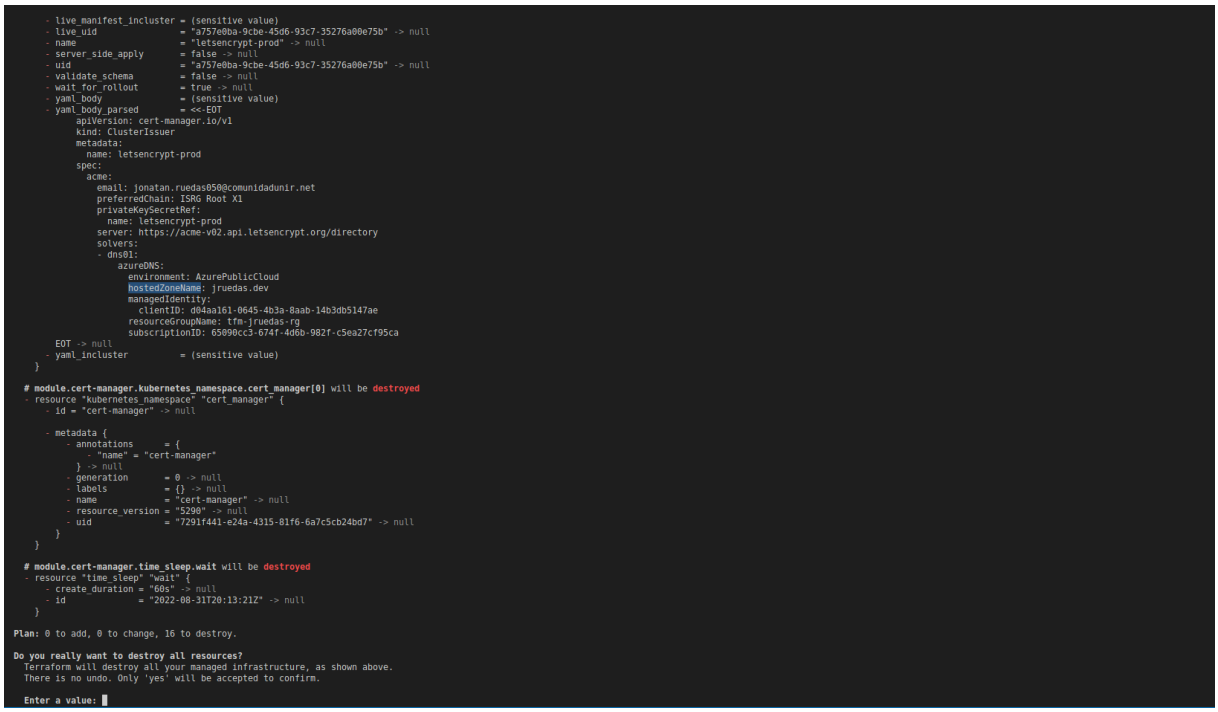
```
                                          - prune    = true
                                          - selfHeal = true
                                        }
                                      - retry       = {
                                          - backoff = {
                                              - duration    = "5s"
                                              - factor      = 2
                                              - maxDuration = "3m0s"
                                            }
                                          - limit   = 4
                                        }
                                      - syncOptions = [
                                          - "CreateNamespace=true",
                                          - "ApplyOutOfSyncOnly=true",
                                          - "PrunePropagationPolicy=background",
                                        ]
                                    }
                                },
                            ]
                          - extraArgs             = [
                              - "--insecure",
                            ]
                        }
                    }
                )
              - version     = "4.10.8"
            },
        ] -> null
      - name                       = "argocd" -> null
      - namespace                  = "argocd" -> null
      - pass_credentials           = false -> null
      - recreate_pods              = false -> null
      - render_subchart_notes      = true -> null
      - replace                    = false -> null
      - repository                 = "https://argoproj.github.io/argo-helm" -> null
      - reset_values               = false -> null
      - reuse_values               = false -> null
      - skip_crds                  = false -> null
      - status                     = "deployed" -> null
      - timeout                    = 300 -> null
      - values                     = [
          - <<-EOT
              server:
                extraArgs:
                  - --insecure
                additionalApplications:
                  - name: root-app
                    namespace: argocd
                    project: default
                    source:
                      path: applications/
                      repoURL: https://github.com/JRuedas/tfm-argocd-apps.git
                      targetRevision: HEAD
                    destination:
                      server: https://kubernetes.default.svc
                      namespace: root-app
                    syncPolicy:
                      automated:
                        prune: true
                        selfHeal: true
                      syncOptions:
                        - CreateNamespace=true
```



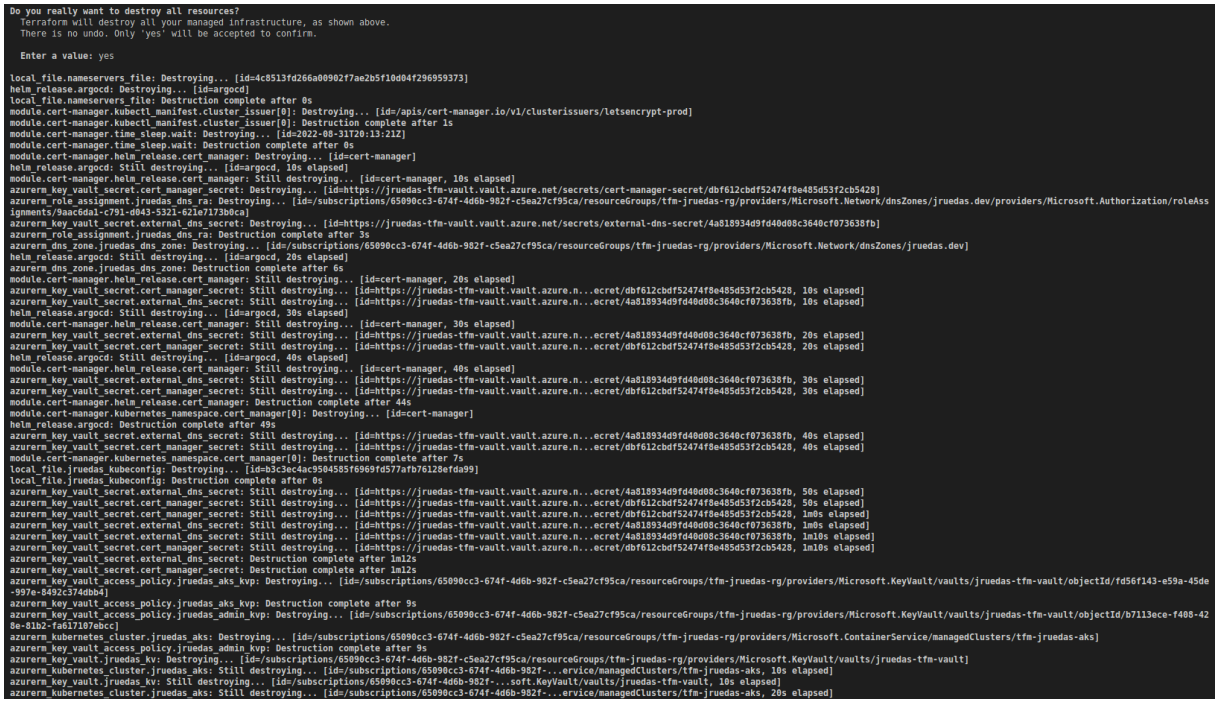
```
                      - ApplyOutOfSyncOnly=true
                      - PrunePropagationPolicy=background
                    retry:
                      limit: 4
                      backoff:
                        duration: 5s
                        factor: 2
                        maxDuration: 3m0s
            EOT,
        ] -> null
      - verify                    = false -> null
      - version                   = "4.10.8" -> null
      - wait                      = true -> null
      - wait_for_jobs             = false -> null
    }

  # local_file.jruedas_kubeconfig will be destroyed
  - resource "local_file" "jruedas_kubeconfig" {
      - content              = (sensitive) -> null
      - directory_permission = "0777" -> null
      - file_permission      = "600" -> null
      - filename             = "/home/jonatan/.kube/aks-tfm.kubeconfig" -> null
      - id                   = "b3c3ec4ac9504585f6969fd577afb76128efda99" -> null
    }

  # local_file.nameservers_file will be destroyed
  - resource "local_file" "nameservers_file" {
      - content              = <<-EOT
            ns1-04.azure-dns.com.
            ns2-04.azure-dns.net.
            ns3-04.azure-dns.org.
            ns4-04.azure-dns.info.
        EOT -> null
      - directory_permission = "0777" -> null
      - file_permission      = "0777" -> null
      - filename             = "nameservers.txt" -> null
      - id                   = "4c8513fd266a00902f7ae2b5f18d84f296959373" -> null
    }

  # module.cert-manager.helm_release.cert_manager will be destroyed
  - resource "helm_release" "cert_manager" {
      - atomic                    = false -> null
      - chart                     = "cert-manager" -> null
      - cleanup_on_fail           = false -> null
      - create_namespace          = false -> null
      - dependency_update         = false -> null
      - disable_crd_hooks         = false -> null
      - disable_openapi_validation = false -> null
      - disable_webhooks          = false -> null
      - force_update              = false -> null
      - id                        = "cert-manager" -> null
      - lint                      = false -> null
      - max_history               = 0 -> null
      - metadata                  = [
          - {
              - app_version = "v1.9.1"
              - chart       = "cert-manager"
              - name        = "cert-manager"
              - namespace   = "cert-manager"
              - revision    = 1
              - values      = jsonencode(
                    {
```



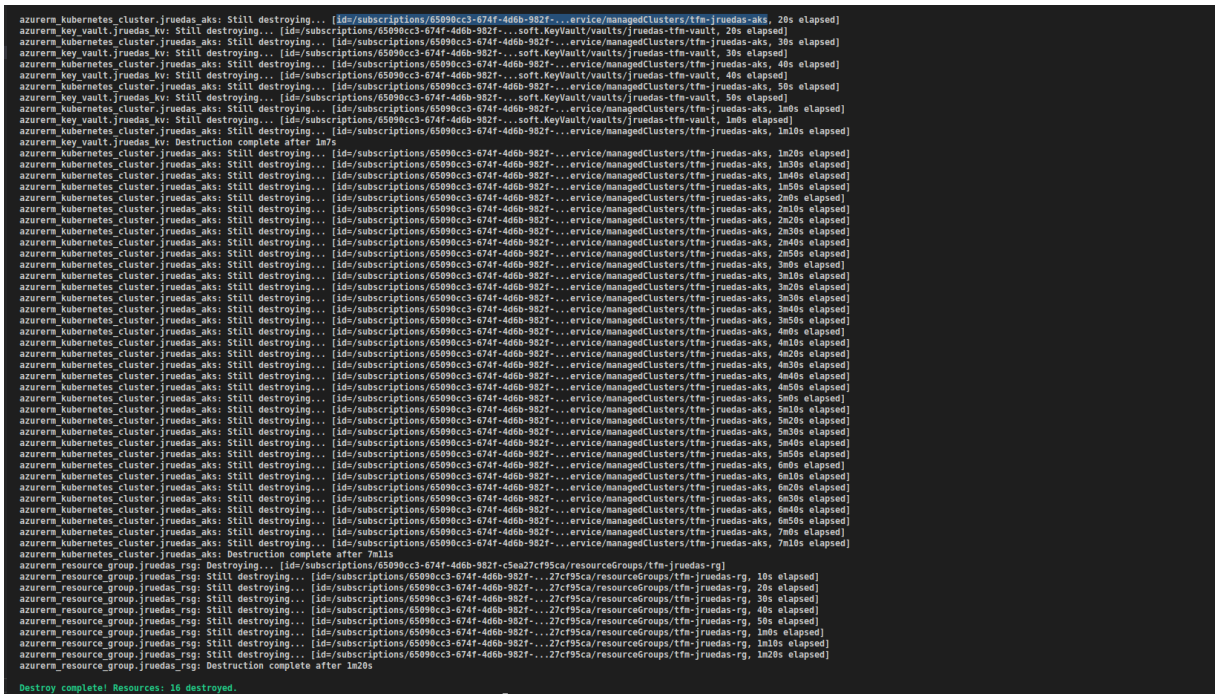
```
                      - installCRDs = true
                    }
                )
              - version     = "v1.9.1"
            },
        ] -> null
      - name                       = "cert-manager" -> null
      - namespace                  = "cert-manager" -> null
      - pass_credentials           = false -> null
      - recreate_pods              = false -> null
      - render_subchart_notes      = true -> null
      - replace                    = false -> null
      - repository                 = "https://charts.jetstack.io" -> null
      - reset_values               = false -> null
      - reuse_values               = false -> null
      - skip_crds                  = false -> null
      - status                     = "deployed" -> null
      - timeout                    = 300 -> null
      - verify                     = false -> null
      - version                    = "v1.9.1" -> null
      - wait                       = true -> null
      - wait_for_jobs              = false -> null

      - set {
          - name  = "installCRDs" -> null
          - value = "true" -> null
        }
    }

  # module.cert-manager.kubectl_manifest.cluster_issuer[0] will be destroyed
  - resource "kubectl_manifest" "cluster_issuer" {
      - api_version             = "cert-manager.io/v1" -> null
      - apply_only              = false -> null
      - force_conflicts         = false -> null
      - force_new               = false -> null
      - id                      = "/apis/cert-manager.io/v1/clusterissuers/letsencrypt-prod" -> null
      - kind                    = "ClusterIssuer" -> null
      - live_manifest_incluster = (sensitive value)
      - live_uid                = "a757e0ba-9cbe-45d6-93c7-35276a00e75b" -> null
      - name                    = "letsencrypt-prod" -> null
      - server_side_apply       = false -> null
      - uid                     = "a757e0ba-9cbe-45d6-93c7-35276a00e75b" -> null
      - validate_schema         = false -> null
      - wait_for_rollout        = true -> null
      - yaml_body               = (sensitive value)
      - yaml_body_parsed        = <<-EOT
            apiVersion: cert-manager.io/v1
            kind: ClusterIssuer
            metadata:
              name: letsencrypt-prod
            spec:
              acme:
                email: jonatan.ruedas850@comunidadunir.net
                preferredChain: ISRG Root X1
                privateKeySecretRef:
                  name: letsencrypt-prod
                server: https://acme-v02.api.letsencrypt.org/directory
                solvers:
                - dns01:
                    azureDNS:
                      environment: AzurePublicCloud
                      hostedZoneName: jruedas.dev
```



```
      - live_manifest_incluster = (sensitive value)
      - live_uid                = "a757e0ba-9cbe-45d6-93c7-35276a00e75b" -> null
      - name                    = "letsencrypt-prod" -> null
      - server_side_apply       = false -> null
      - uid                     = "a757e0ba-9cbe-45d6-93c7-35276a00e75b" -> null
      - validate_schema         = false -> null
      - wait_for_rollout        = true -> null
      - yaml_body               = (sensitive value)
      - yaml_body_parsed        = <<-EOT
            apiVersion: cert-manager.io/v1
            kind: ClusterIssuer
            metadata:
              name: letsencrypt-prod
            spec:
              acme:
                email: jonatan.ruedas050@comunidadunir.net
                preferredChain: ISRG Root X1
                privateKeySecretRef:
                  name: letsencrypt-prod
                server: https://acme-v02.api.letsencrypt.org/directory
                solvers:
                - dns01:
                    azureDNS:
                      environment: AzurePublicCloud
                      hostedZoneName: jruedas.dev
                      managedIdentity:
                        clientID: d04aa161-0645-4b3a-8aab-14b3db5147ae
                      resourceGroupName: tfm-jruedas-rg
                      subscriptionID: 65090cc3-674f-4d6b-982f-c5ea27cf95ca
        EOT -> null
      - yaml_incluster          = (sensitive value)
    }

  # module.cert-manager.kubernetes_namespace.cert_manager[0] will be destroyed
  - resource "kubernetes_namespace" "cert_manager" {
      - id = "cert-manager" -> null

      - metadata {
          - annotations      = {
              - "name" = "cert-manager"
            } -> null
          - generation       = 0 -> null
          - labels           = {} -> null
          - name             = "cert-manager" -> null
          - resource_version = "5290" -> null
          - uid              = "7291f441-e24a-4315-81f6-6a7c5cb24bd7" -> null
        }
    }

  # module.cert-manager.time_sleep.wait will be destroyed
  - resource "time_sleep" "wait" {
      - create_duration = "60s" -> null
      - id              = "2022-08-31T20:13:21Z" -> null
    }

Plan: 0 to add, 0 to change, 16 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value:
```



```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

local_file.nameservers_file: Destroying... [id=4c8513fd266a00902f7ae2b5f10d04f296959373]
helm_release.argocd: Destroying... [id=argocd]
local_file.nameservers_file: Destruction complete after 0s
module.cert-manager.kubectl_manifest.cluster_issuer[0]: Destroying... [id=/apis/cert-manager.io/v1/clusterissuers/letsencrypt-prod]
module.cert-manager.kubectl_manifest.cluster_issuer[0]: Destruction complete after 1s
module.cert-manager.time_sleep.wait: Destroying... [id=2022-08-31T20:13:21Z]
module.cert-manager.time_sleep.wait: Destruction complete after 0s
module.cert-manager.helm_release.cert_manager: Destroying... [id=cert-manager]
helm_release.argocd: Still destroying... [id=argocd, 10s elapsed]
module.cert-manager.helm_release.cert_manager: Still destroying... [id=cert-manager, 10s elapsed]
azurerm_key_vault_secret.cert_manager_secret: Destroying... [id=https://jruedas-tfm-vault.vault.azure.net/secrets/cert-manager-secret/dbf612cbdf52474f8e485d53f2cb5428]
azurerm_role_assignment.jruedas_dns_ra: Destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.Network/dnsZones/jruedas.dev/providers/Microsoft.Authorization/roleAss
ignments/9aac6da1-c791-d043-5321-621e7173b0ca]
azurerm_key_vault_secret.external_dns_secret: Destroying... [id=https://jruedas-tfm-vault.vault.azure.net/secrets/external-dns-secret/4a818934d9fd40d08c3640cf073638fb]
azurerm_role_assignment.jruedas_dns_ra: Destruction complete after 3s
azurerm_dns_zone.jruedas_dns_zone: Destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.Network/dnsZones/jruedas.dev]
helm_release.argocd: Still destroying... [id=argocd, 20s elapsed]
azurerm_dns_zone.jruedas_dns_zone: Destruction complete after 6s
module.cert-manager.helm_release.cert_manager: Still destroying... [id=cert-manager, 20s elapsed]
azurerm_key_vault_secret.cert_manager_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/dbf612cbdf52474f8e485d53f2cb5428, 10s elapsed]
azurerm_key_vault_secret.external_dns_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/4a818934d9fd40d08c3640cf073638fb, 10s elapsed]
helm_release.argocd: Still destroying... [id=argocd, 30s elapsed]
module.cert-manager.helm_release.cert_manager: Still destroying... [id=cert-manager, 30s elapsed]
azurerm_key_vault_secret.external_dns_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/4a818934d9fd40d08c3640cf073638fb, 20s elapsed]
azurerm_key_vault_secret.cert_manager_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/dbf612cbdf52474f8e485d53f2cb5428, 20s elapsed]
helm_release.argocd: Still destroying... [id=argocd, 40s elapsed]
module.cert-manager.helm_release.cert_manager: Still destroying... [id=cert-manager, 40s elapsed]
azurerm_key_vault_secret.external_dns_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/4a818934d9fd40d08c3640cf073638fb, 30s elapsed]
azurerm_key_vault_secret.cert_manager_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/dbf612cbdf52474f8e485d53f2cb5428, 30s elapsed]
module.cert-manager.helm_release.cert_manager: Destruction complete after 44s
module.cert-manager.kubernetes_namespace.cert_manager[0]: Destroying... [id=cert-manager]
helm_release.argocd: Destruction complete after 49s
azurerm_key_vault_secret.external_dns_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/4a818934d9fd40d08c3640cf073638fb, 40s elapsed]
azurerm_key_vault_secret.cert_manager_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/dbf612cbdf52474f8e485d53f2cb5428, 40s elapsed]
module.cert-manager.kubernetes_namespace.cert_manager[0]: Destruction complete after 7s
local_file.jruedas_kubeconfig: Destroying... [id=b3c3ec4ac9504585f6969fd577afb76128efda99]
local_file.jruedas_kubeconfig: Destruction complete after 0s
azurerm_key_vault_secret.external_dns_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/4a818934d9fd40d08c3640cf073638fb, 50s elapsed]
azurerm_key_vault_secret.cert_manager_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/dbf612cbdf52474f8e485d53f2cb5428, 50s elapsed]
azurerm_key_vault_secret.cert_manager_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/dbf612cbdf52474f8e485d53f2cb5428, 1m0s elapsed]
azurerm_key_vault_secret.external_dns_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/4a818934d9fd40d08c3640cf073638fb, 1m0s elapsed]
azurerm_key_vault_secret.external_dns_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/4a818934d9fd40d08c3640cf073638fb, 1m10s elapsed]
azurerm_key_vault_secret.cert_manager_secret: Still destroying... [id=https://jruedas-tfm-vault.vault.azure.n...ecret/dbf612cbdf52474f8e485d53f2cb5428, 1m10s elapsed]
azurerm_key_vault_secret.external_dns_secret: Destruction complete after 1m12s
azurerm_key_vault_secret.cert_manager_secret: Destruction complete after 1m12s
azurerm_key_vault_access_policy.jruedas_aks_kvp: Destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault/objectId/fd56f143-e59a-45de
-997e-8492c374dbb4]
azurerm_key_vault_access_policy.jruedas_aks_kvp: Destruction complete after 9s
azurerm_key_vault_access_policy.jruedas_admin_kvp: Destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault/objectId/b7113ece-f408-42
8e-81b2-fa617107e0cc]
azurerm_kubernetes_cluster.jruedas_aks: Destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.ContainerService/managedClusters/tfm-jruedas-aks]
azurerm_key_vault_access_policy.jruedas_admin_kvp: Destruction complete after 9s
azurerm_key_vault.jruedas_kv: Destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg/providers/Microsoft.KeyVault/vaults/jruedas-tfm-vault]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 10s elapsed]
azurerm_key_vault.jruedas_kv: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...soft.KeyVault/vaults/jruedas-tfm-vault, 10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 20s elapsed]
```


azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 20s elapsed]
azurerm_key_vault.jruedas_kv: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...soft.KeyVault/vaults/jruedas-tfm-vault, 20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 30s elapsed]
azurerm_key_vault.jruedas_kv: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...soft.KeyVault/vaults/jruedas-tfm-vault, 30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 40s elapsed]
azurerm_key_vault.jruedas_kv: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...soft.KeyVault/vaults/jruedas-tfm-vault, 40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 50s elapsed]
azurerm_key_vault.jruedas_kv: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...soft.KeyVault/vaults/jruedas-tfm-vault, 50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 1m0s elapsed]
azurerm_key_vault.jruedas_kv: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...soft.KeyVault/vaults/jruedas-tfm-vault, 1m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 1m10s elapsed]
azurerm_key_vault.jruedas_kv: Destruction complete after 1m7s
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 1m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 1m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 1m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 1m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 2m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 2m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 2m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 2m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 2m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 2m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 3m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 3m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 3m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 3m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 3m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 3m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 4m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 4m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 4m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 4m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 4m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 4m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 5m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 5m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 5m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 5m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 5m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 5m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 6m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 6m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 6m20s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 6m30s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 6m40s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 6m50s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 7m0s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...ervice/managedClusters/tfm-jruedas-aks, 7m10s elapsed]
azurerm_kubernetes_cluster.jruedas_aks: Destruction complete after 7m11s
azurerm_resource_group.jruedas_rsg: Destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-c5ea27cf95ca/resourceGroups/tfm-jruedas-rg]
azurerm_resource_group.jruedas_rsg: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...27cf95ca/resourceGroups/tfm-jruedas-rg, 10s elapsed]
azurerm_resource_group.jruedas_rsg: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...27cf95ca/resourceGroups/tfm-jruedas-rg, 20s elapsed]
azurerm_resource_group.jruedas_rsg: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...27cf95ca/resourceGroups/tfm-jruedas-rg, 30s elapsed]
azurerm_resource_group.jruedas_rsg: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...27cf95ca/resourceGroups/tfm-jruedas-rg, 40s elapsed]
azurerm_resource_group.jruedas_rsg: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...27cf95ca/resourceGroups/tfm-jruedas-rg, 50s elapsed]
azurerm_resource_group.jruedas_rsg: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...27cf95ca/resourceGroups/tfm-jruedas-rg, 1m0s elapsed]
azurerm_resource_group.jruedas_rsg: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...27cf95ca/resourceGroups/tfm-jruedas-rg, 1m10s elapsed]
azurerm_resource_group.jruedas_rsg: Still destroying... [id=/subscriptions/65090cc3-674f-4d6b-982f-...27cf95ca/resourceGroups/tfm-jruedas-rg, 1m20s elapsed]
azurerm_resource_group.jruedas_rsg: Destruction complete after 1m20s
Destroy complete! Resources: 16 destroyed.

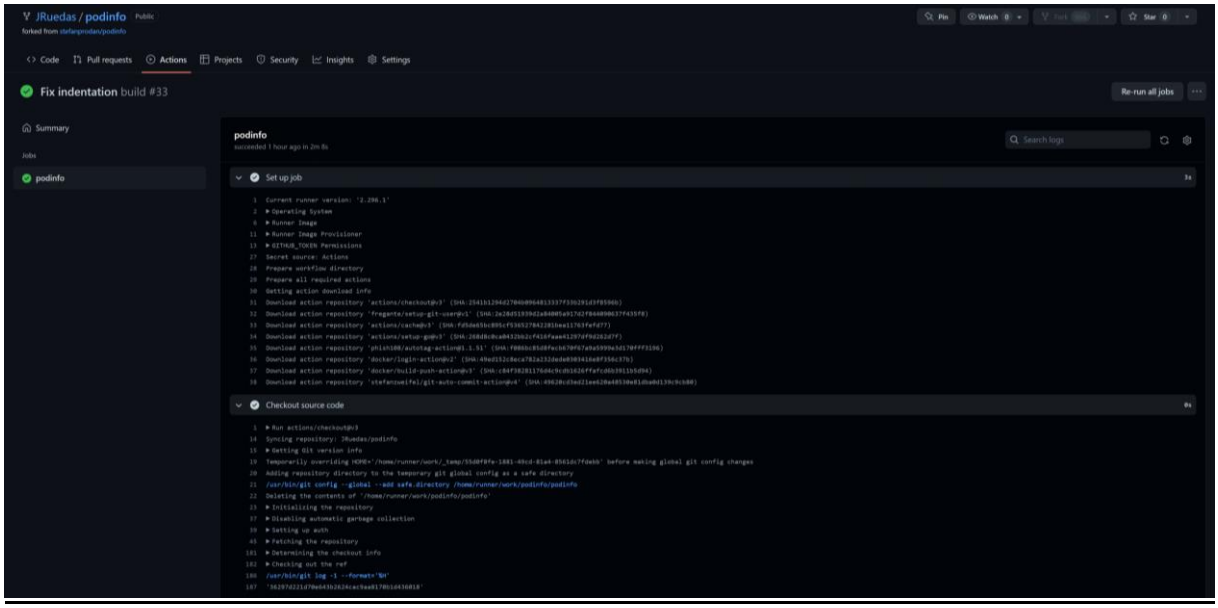
Anexo B. CI/CD: GitHub Actions

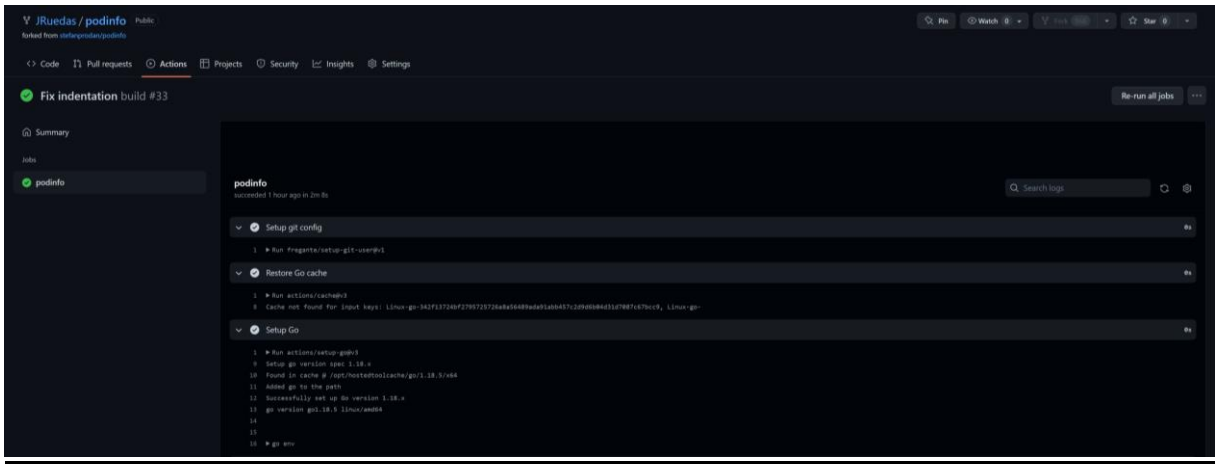
B-1. Fichero de configuración de GitHub Actions

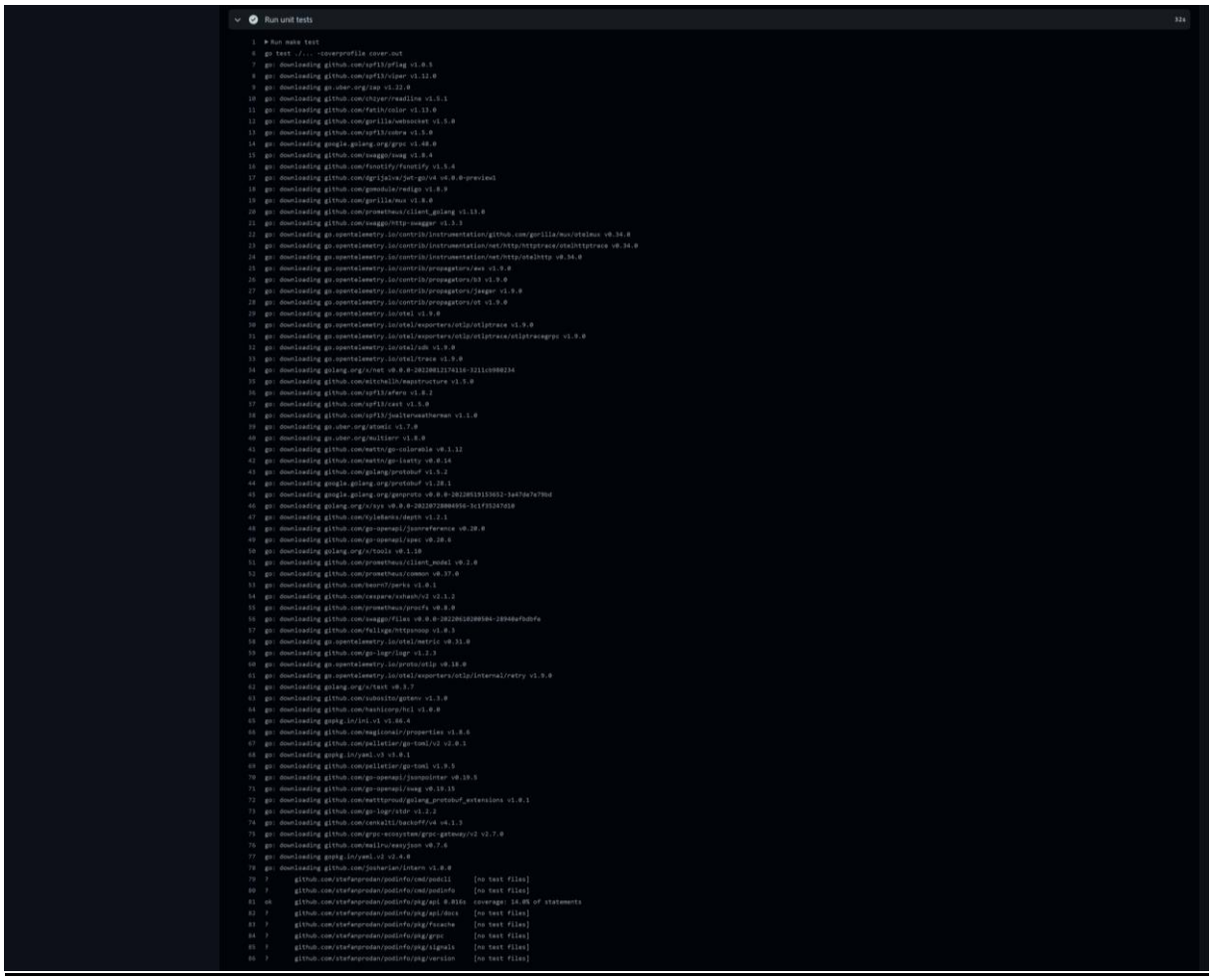
podinfo > .github > workflows > ! build.yml

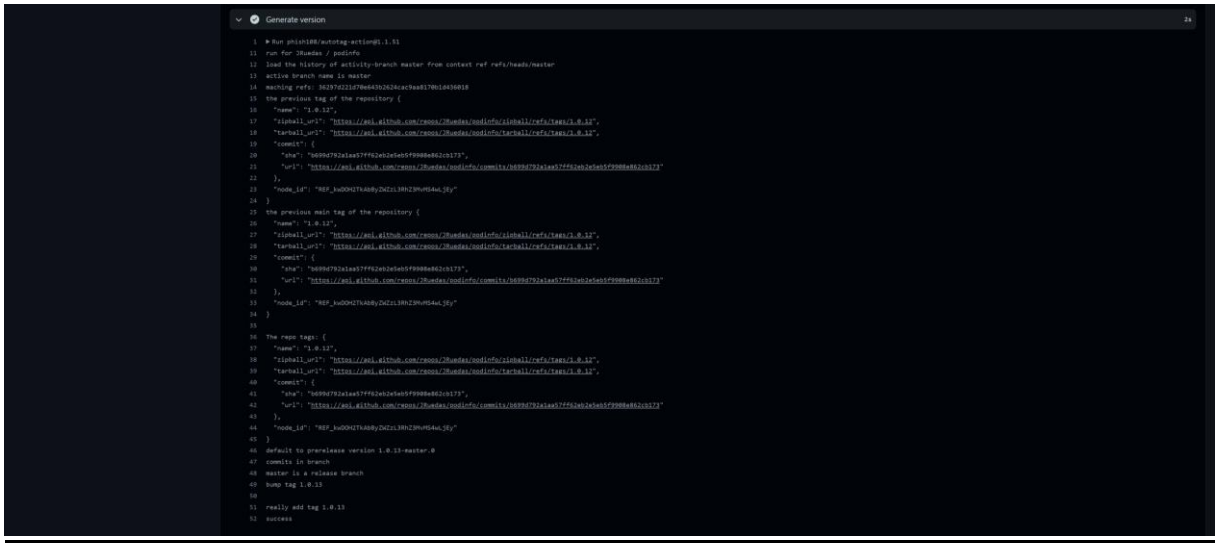
```yaml
name: build

on:
  push:
    branches:
      - 'master'
    paths-ignore:
      - 'pkg/version/version.go'

jobs:
  podinfo:
    env:
      IMAGE: ${{ secrets.DOCKER_USERNAME }}/podinfo
    runs-on: ubuntu-latest
    steps:
    - name: Checkout source code
      uses: actions/checkout@v3

    - name: Setup git config
      uses: fregante/setup-git-user@v1

    - name: Restore Go cache
      uses: actions/cache@v3
      with:
        path: ~/go/pkg/mod
        key: ${{ runner.os }}-go-${{ hashFiles('**/go.sum') }}
        restore-keys: ${{ runner.os }}-go-

    - name: Setup Go
      uses: actions/setup-go@v3
      with:
        go-version: 1.18.x

    - name: Run unit tests
      run: make test

    - name: Generate version
      id: version
      uses: phish108/autotag-action@1.1.51
      with:
        github-token: ${{ secrets.GH_TOKEN}}

    - name: Update app version
      run: |
          version=$(cat ./pkg/version/version.go | grep 'VERSION' pkg/version/version.go | awk '{ print $$4 }' | tr -d '"' | awk '{print $4}')
          sed -i "s/$version/${{ steps.version.outputs.new-tag }}/" ./pkg/version/version.go

    - name: Login to Docker Hub
      uses: docker/login-action@v2
      with:
        username: ${{ secrets.DOCKER_USERNAME }}
        password: ${{ secrets.DOCKER_PASSWORD }}

    - name: Build & Push Docker image
      id: docker_build
      uses: docker/build-push-action@v3
      with:
        context: .
        push: true
        tags: |
          ${{ env.IMAGE }}:${{ steps.version.outputs.new-tag }}
          ${{ env.IMAGE }}:latest
        labels: ${{ steps.meta.outputs.labels }}

    - name: Release app new version
      uses: stefanzweifel/git-auto-commit-action@v4
      with:
        commit_message: Version updated ${{ steps.version.outputs.new-tag }}
        file_pattern: ./pkg/version/version.go

    - name: Checkout configuration
      uses: actions/checkout@v3
      with:
        repository: JRuedas/tfm-argocd-apps
        path: ./tfm-argocd-apps
        submodules: recursive
        token: ${{ secrets.GH_TOKEN}}

    - name: Update configuration version
      run: |
          version=$(cat ./tfm-argocd-apps/charts/jruedas-podinfo/Chart.yaml | grep -m 1 appVersion: | awk '{print $2}')
          sed -i "0,/tag: $version/{s/tag: $version/tag: ${{ steps.version.outputs.new-tag }}/}" ./tfm-argocd-apps/charts/jruedas-podinfo/values.yaml
          sed -i "s/appVersion: $version/appVersion: ${{ steps.version.outputs.new-tag }}/" ./tfm-argocd-apps/charts/jruedas-podinfo/Chart.yaml

    - name: Release configuration new version
      uses: stefanzweifel/git-auto-commit-action@v4
      with:
        repository: ./tfm-argocd-apps
        commit_message: Version updated ${{ steps.version.outputs.new-tag }}
        file_pattern: ./charts/jruedas-podinfo/values.yaml ./charts/jruedas-podinfo/Chart.yaml
```

B-2. Ejecución de GitHub Actions

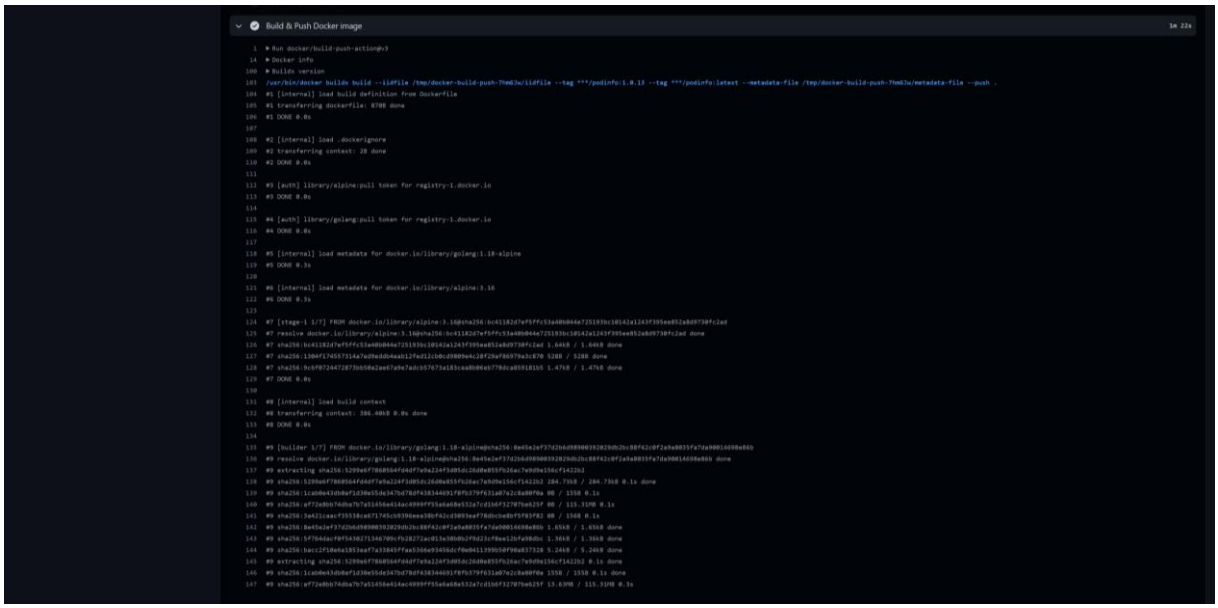


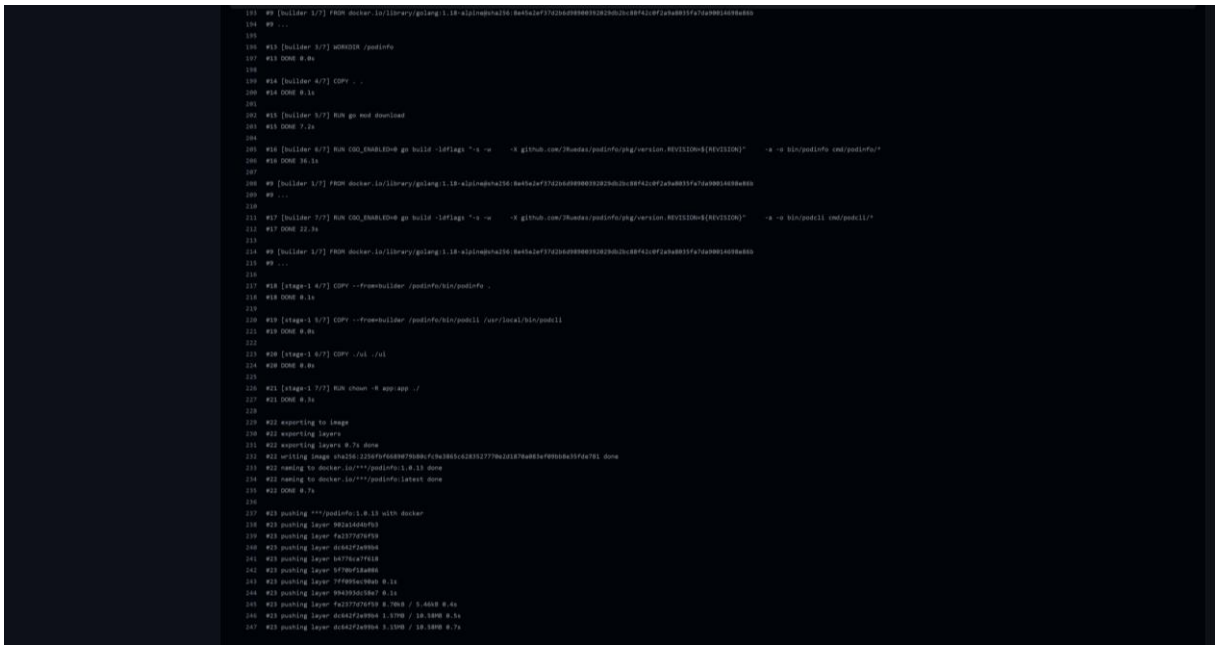


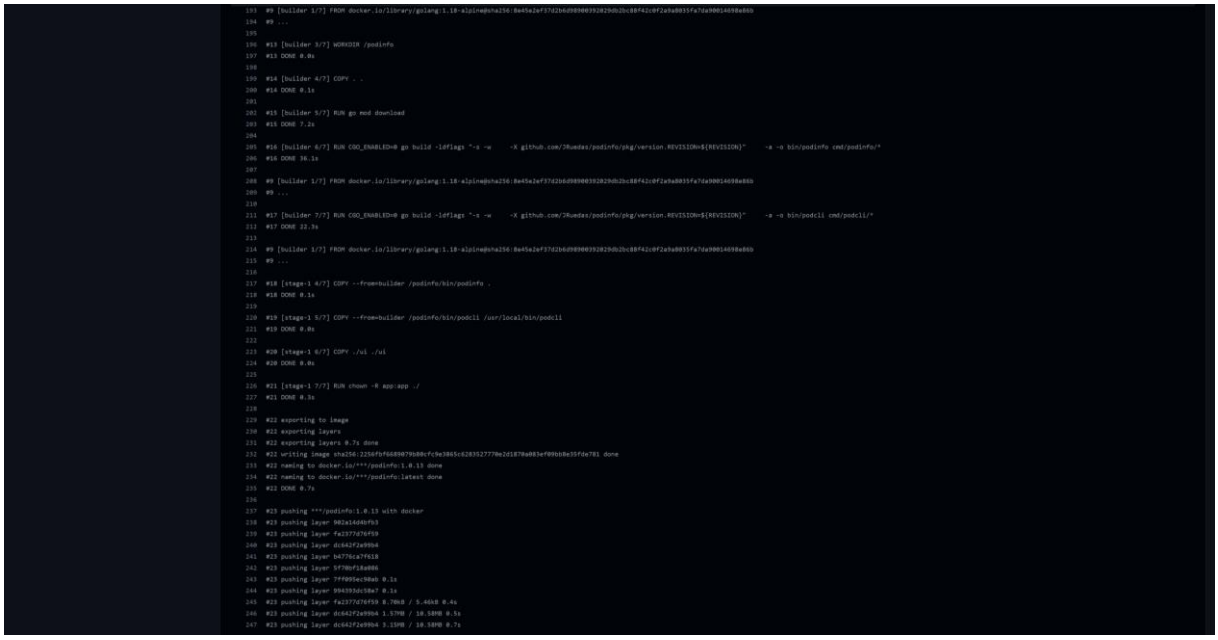

Run unit tests
32s
Run make test
go test ./... -coverprofile cover.out
go: downloading github.com/spf13/pflag v1.0.5
go: downloading github.com/spf13/viper v1.12.0
go: downloading go.uber.org/zap v1.22.0
go: downloading github.com/chzyer/readline v1.5.1
go: downloading github.com/fatih/color v1.13.0
go: downloading github.com/gorilla/websocket v1.5.0
go: downloading github.com/spf13/cobra v1.5.0
go: downloading google.golang.org/grpc v1.48.0
go: downloading github.com/swaggo/swag v1.8.4
go: downloading github.com/fsnotify/fsnotify v1.5.4
go: downloading github.com/grpc-ecosystem/grpc-gateway/v2 v2.0.0-preview1
go: downloading github.com/gomodule/redigo v1.8.9
go: downloading github.com/gorilla/mux v1.8.0
go: downloading github.com/prometheus/client_golang v1.13.0
go: downloading github.com/swaggo/http-swagger v1.3.3
go: downloading go.opentelemetry.io/contrib/instrumentation/github.com/gorilla/mux/otelmux v0.34.0
go: downloading go.opentelemetry.io/contrib/instrumentation/net/http/httptrace/otelhttptrace v0.34.0
go: downloading go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp v0.34.0
go: downloading go.opentelemetry.io/contrib/propagators/aws v1.9.0
go: downloading go.opentelemetry.io/contrib/propagators/b3 v1.9.0
go: downloading go.opentelemetry.io/contrib/propagators/jaeger v1.9.0
go: downloading go.opentelemetry.io/contrib/propagators/ot v1.9.0
go: downloading go.opentelemetry.io/otel v1.9.0
go: downloading go.opentelemetry.io/otel/exporters/otlp/otlptrace v1.9.0
go: downloading go.opentelemetry.io/otel/exporters/otlp/otlptrace/otlptracegrpc v1.9.0
go: downloading go.opentelemetry.io/otel/sdk v1.9.0
go: downloading go.opentelemetry.io/otel/trace v1.9.0
go: downloading golang.org/x/net v0.0.0-20220812174116-3211cb980234
go: downloading github.com/mitchellh/mapstructure v1.5.0
go: downloading github.com/spf13/afero v1.8.2
go: downloading github.com/spf13/cast v1.5.0
go: downloading github.com/spf13/jwalterweatherman v1.1.0
go: downloading go.uber.org/atomic v1.7.0
go: downloading go.uber.org/multierr v1.8.0
go: downloading github.com/mattn/go-colorable v0.1.12
go: downloading github.com/mattn/go-isatty v0.0.14
go: downloading github.com/golang/protobuf v1.5.2
go: downloading google.golang.org/protobuf v1.28.1
go: downloading google.golang.org/genproto v0.0.0-20220519153652-3a47de7e79bd
go: downloading golang.org/x/sys v0.0.0-20220728004956-3c1f35247d10
go: downloading github.com/kyleconroy/doubleclick v1.2.1
go: downloading github.com/go-openapi/jsonreference v0.20.0
go: downloading github.com/go-openapi/spec v0.20.6
go: downloading golang.org/x/tools v0.1.10
go: downloading github.com/prometheus/client_model v0.2.0
go: downloading github.com/prometheus/common v0.37.0
go: downloading github.com/beorn7/perks v1.0.1
go: downloading github.com/cespare/xxhash/v2 v2.1.2
go: downloading github.com/prometheus/procfs v0.8.0
go: downloading github.com/swaggo/files v0.0.0-20220610200504-28940afbdbfe
go: downloading github.com/felixge/httpsnoop v1.0.3
go: downloading go.opentelemetry.io/otel/metric v0.31.0
go: downloading github.com/go-logr/logr v1.2.3
go: downloading go.opentelemetry.io/proto/otlp v0.18.0
go: downloading go.opentelemetry.io/otel/exporters/otlp/internal/retry v1.9.0
go: downloading golang.org/x/text v0.3.7
go: downloading github.com/subosito/gotenv v1.3.0
go: downloading github.com/hashicorp/hcl v1.0.0
go: downloading gopkg.in/ini.v1 v1.66.6
go: downloading github.com/magiconair/properties v1.8.6
go: downloading github.com/pelletier/go-toml/v2 v2.0.1
go: downloading gopkg.in/yaml.v3 v3.0.1
go: downloading github.com/pelletier/go-toml v1.9.5
go: downloading github.com/go-openapi/jsonpointer v0.19.5
go: downloading github.com/go-openapi/swag v0.19.15
go: downloading github.com/mattbaird/jsonpatch v0.0.0-20171005235357-81af80346b1a
go: downloading github.com/matttproud/golang_protobuf_extensions v1.0.1
go: downloading github.com/go-logr/stdr v1.2.2
go: downloading github.com/cenkalti/backoff/v4 v4.1.3
go: downloading github.com/grpc-ecosystem/grpc-gateway/v2 v2.7.0
go: downloading github.com/mailru/easyjson v0.7.6
go: downloading gopkg.in/yaml.v2 v2.4.0
go: downloading github.com/josharian/intern v1.0.0
? github.com/stefanprodan/podinfo/cmd/podcli [no test files]
? github.com/stefanprodan/podinfo/cmd/podinfo [no test files]
ok github.com/stefanprodan/podinfo/pkg/api 0.016s coverage: 14.0% of statements
? github.com/stefanprodan/podinfo/pkg/api/docs [no test files]
? github.com/stefanprodan/podinfo/pkg/fscache [no test files]
? github.com/stefanprodan/podinfo/pkg/grpc [no test files]
? github.com/stefanprodan/podinfo/pkg/signals [no test files]
? github.com/stefanprodan/podinfo/pkg/version [no test files]

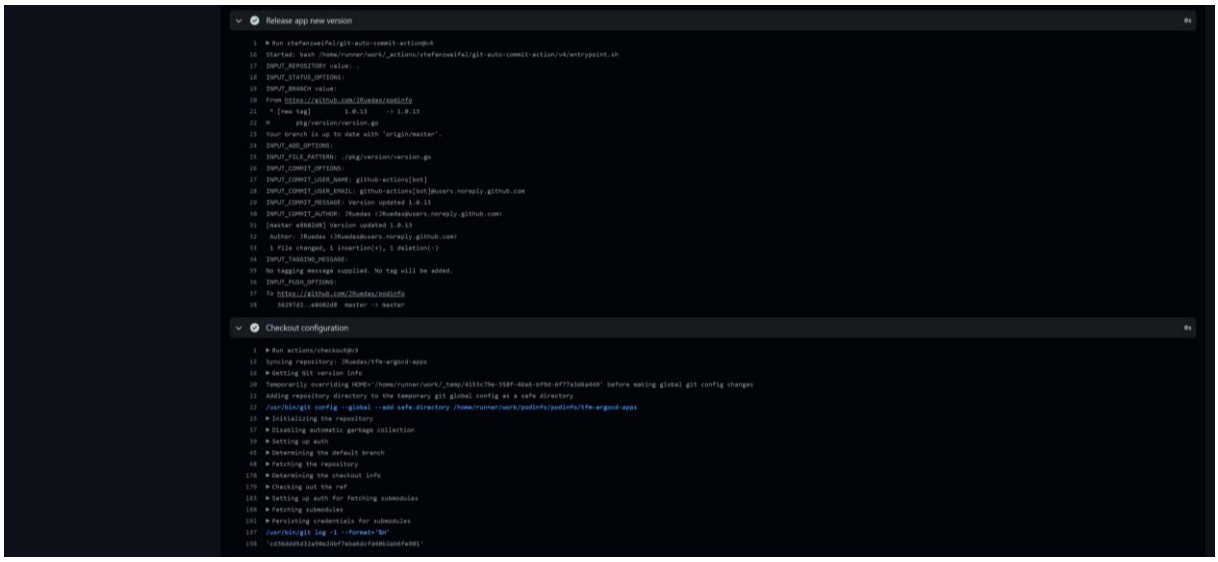

Generate version
2s
Run phish108/autotag-action@1.1.51
run for JRuedas / podinfo
load the history of activity-branch master from context ref refs/heads/master
active branch name is master
matching refs: 362376221bf76e643b2626cac9aa8170b1d436018
the previous tag of the repository {
"name": "1.0.12",
"zipball_url": "https://api.github.com/repos/JRuedas/podinfo/zipball/refs/tags/1.0.12",
"tarball_url": "https://api.github.com/repos/JRuedas/podinfo/tarball/refs/tags/1.0.12",
"commit": {
"sha": "b699d792a1aa57ff62eb2e5eb5f9908e862cb173",
"url": "https://api.github.com/repos/JRuedas/podinfo/commits/b699d792a1aa57ff62eb2e5eb5f9908e862cb173"
},
"node_id": "REF_kwDOH2TkA0ByZWZzL3RhZ3MvMS4wLjEy"
}
the previous main tag of the repository {
"name": "1.0.12",
"zipball_url": "https://api.github.com/repos/JRuedas/podinfo/zipball/refs/tags/1.0.12",
"tarball_url": "https://api.github.com/repos/JRuedas/podinfo/tarball/refs/tags/1.0.12",
"commit": {
"sha": "b699d792a1aa57ff62eb2e5eb5f9908e862cb173",
"url": "https://api.github.com/repos/JRuedas/podinfo/commits/b699d792a1aa57ff62eb2e5eb5f9908e862cb173"
},
"node_id": "REF_kwDOH2TkA0ByZWZzL3RhZ3MvMS4wLjEy"
}
The repo tags: {
"name": "1.0.12",
"zipball_url": "https://api.github.com/repos/JRuedas/podinfo/zipball/refs/tags/1.0.12",
"tarball_url": "https://api.github.com/repos/JRuedas/podinfo/tarball/refs/tags/1.0.12",
"commit": {
"sha": "b699d792a1aa57ff62eb2e5eb5f9908e862cb173",
"url": "https://api.github.com/repos/JRuedas/podinfo/commits/b699d792a1aa57ff62eb2e5eb5f9908e862cb173"
},
"node_id": "REF_kwDOH2TkA0ByZWZzL3RhZ3MvMS4wLjEy"
}
default to prerelease version 1.0.13-master.0
commits in branch
master is a release branch
bump tag 1.0.13
really add tag 1.0.13
success

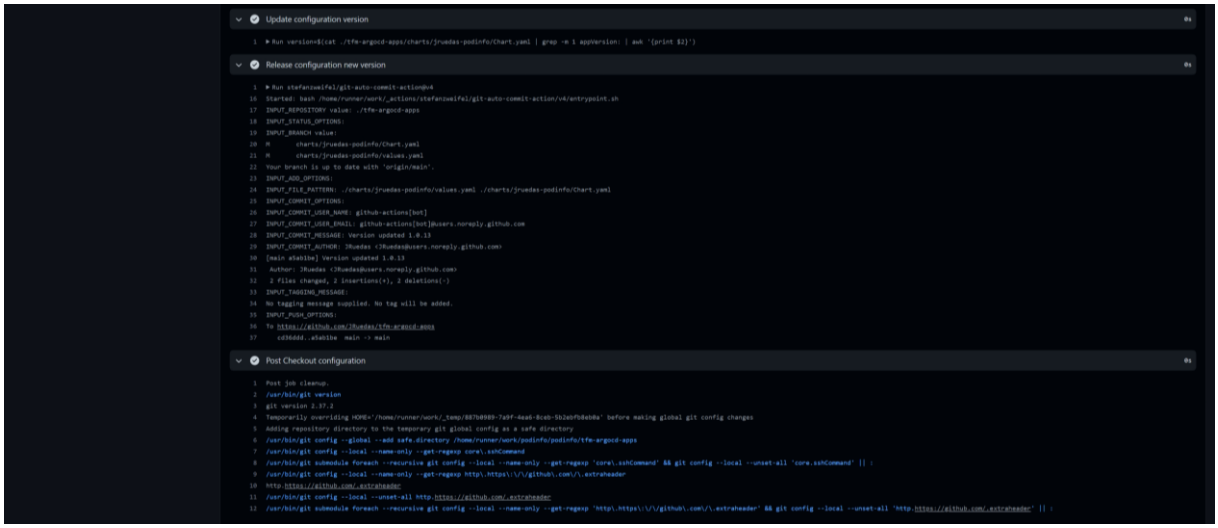

Update app version
0s
Run version=$(cat ./pkg/version/version.go | grep "VERSION" pkg/version/version.go | awk '{ print $$4 }' | tr -d '"' | awk '{print $4}')
Login to Docker Hub
1s
Run docker/login-action@v2
Logging into Docker Hub...
Login Succeeded!

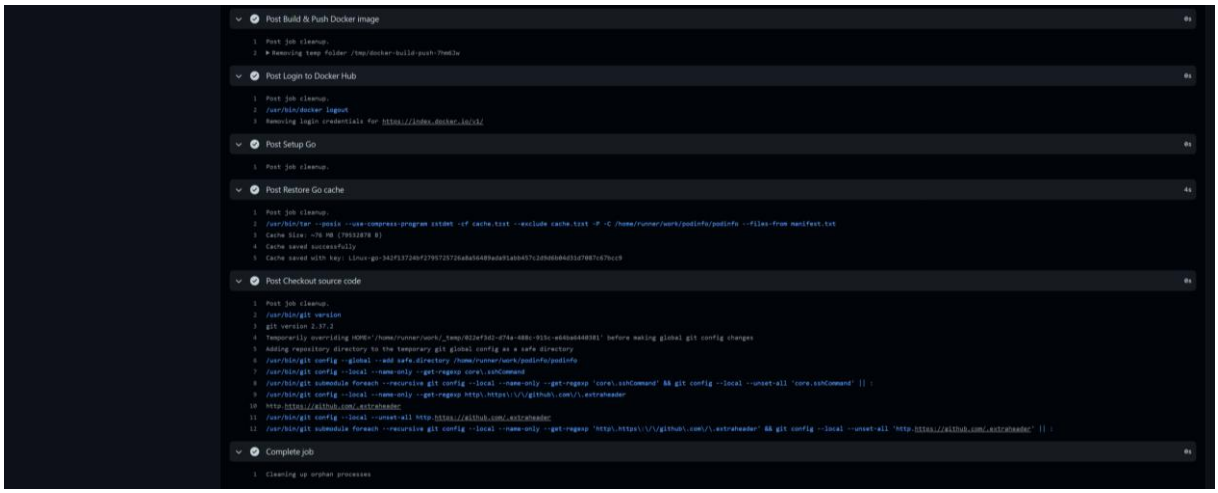

Build & Push Docker image
1m 22s





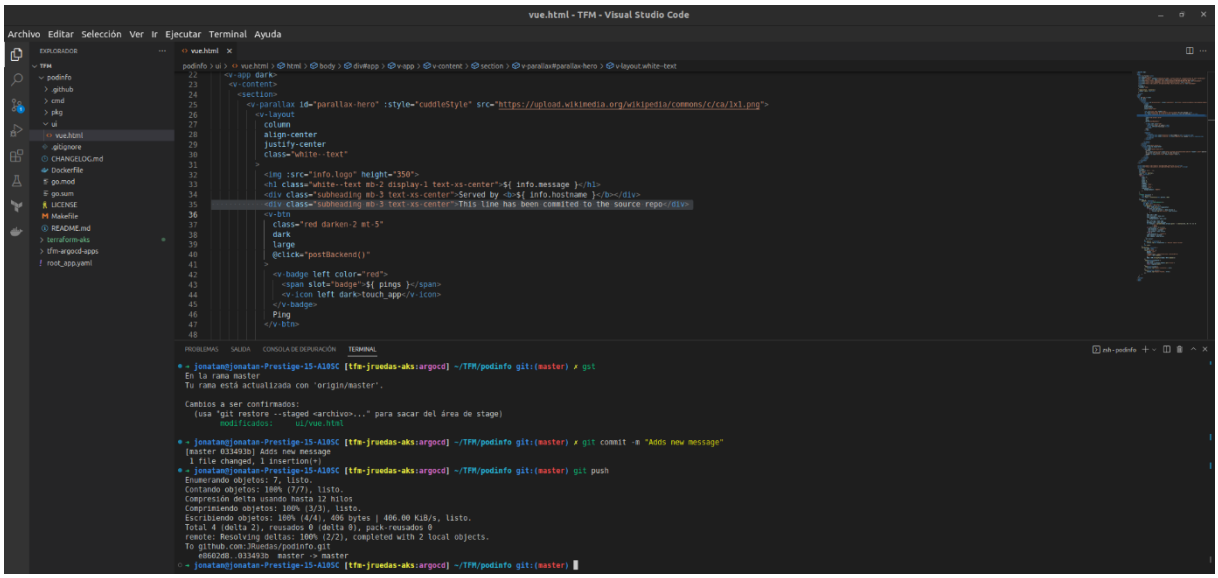

Release app new version
Checkout configuration

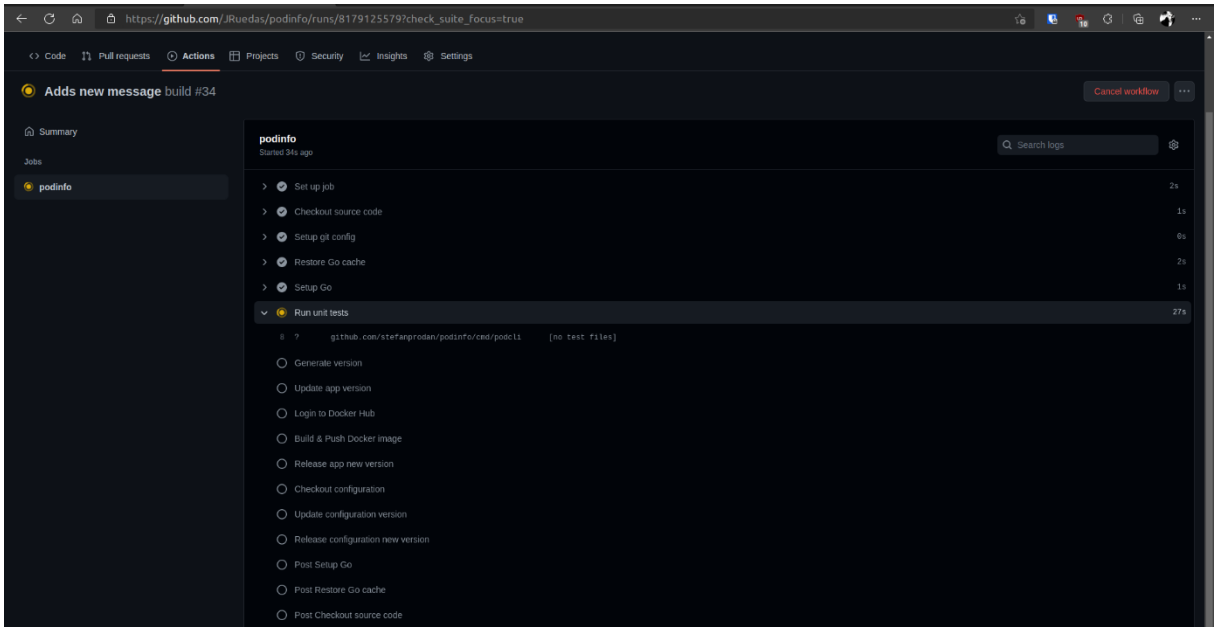

Update configuration version
Release configuration new version
Post Checkout configuration

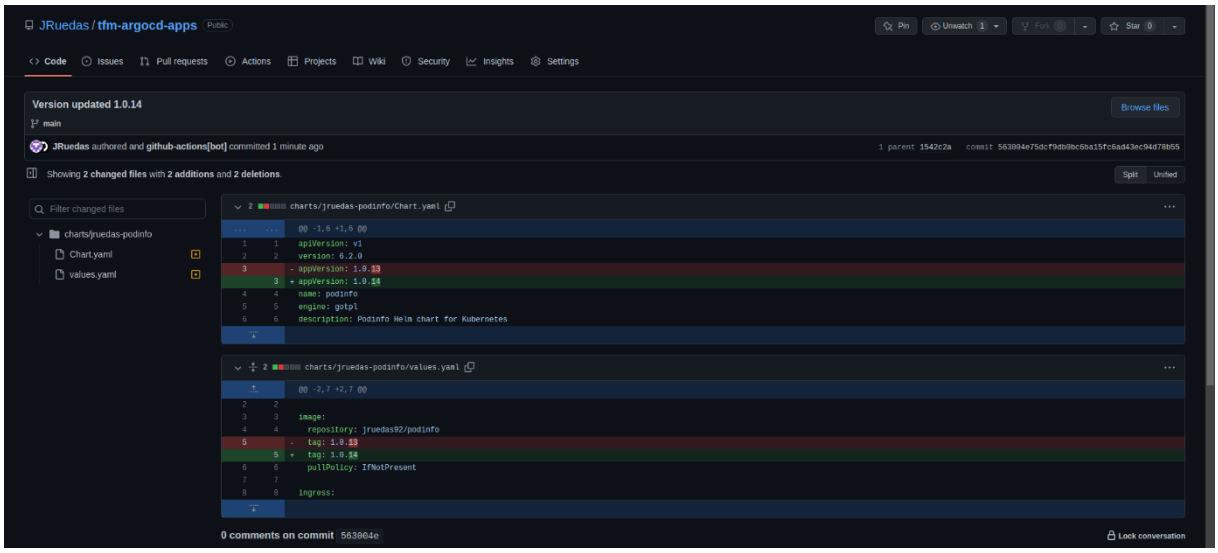

Post Build & Push Docker image
Post Login to Docker Hub
Post Setup Go
Post Restore Go cache
Post Checkout source code
Complete job

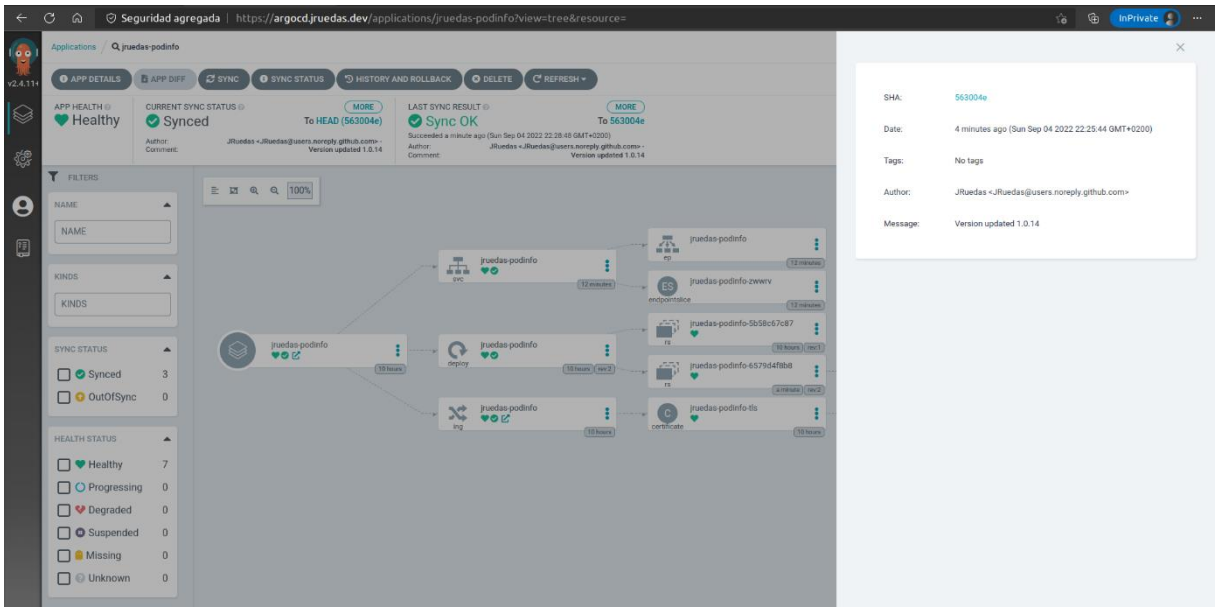
Anexo C. Ejemplo despliegue automático nueva versión

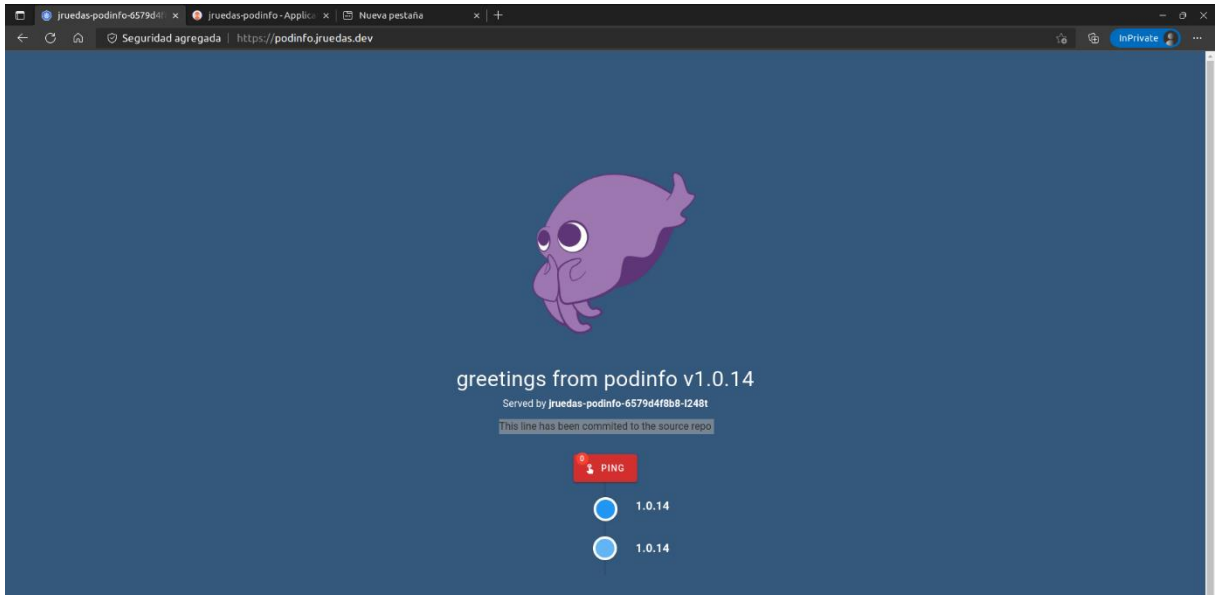
C-1. Ejemplo de despliegue automático de nueva versión



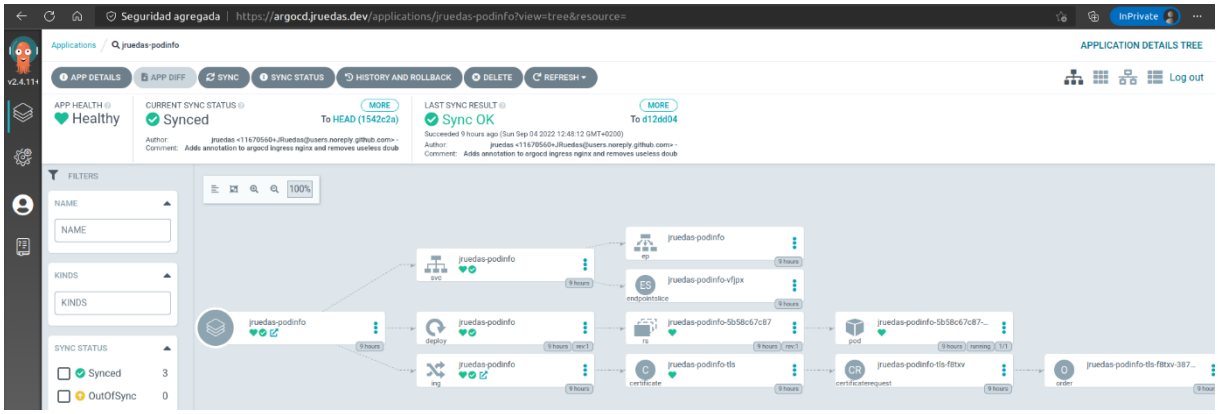


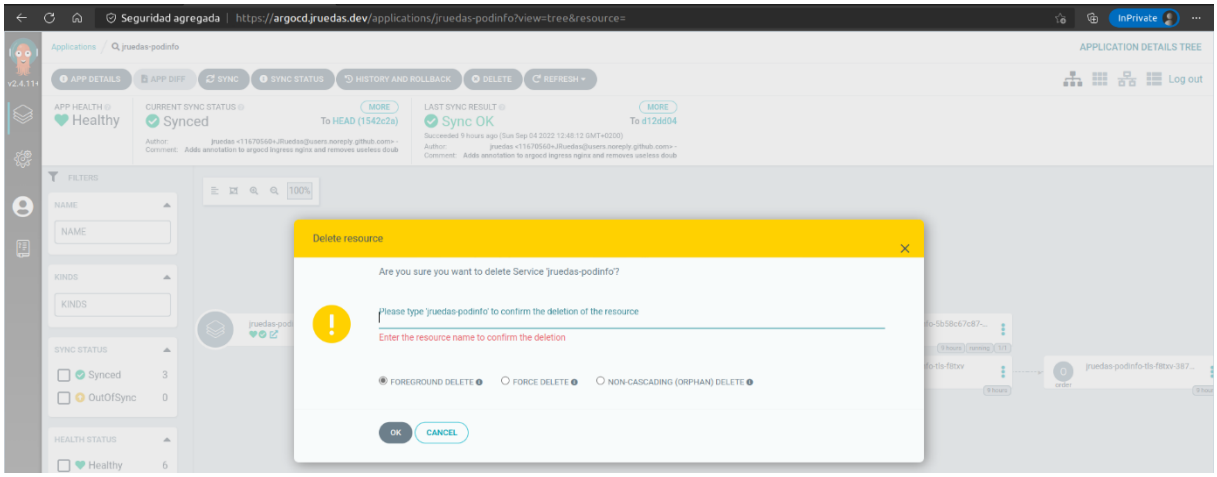

JRuedas / tfm-argocd-apps
Version updated 1.0.14
JRuedas authored and github-actions[bot] committed 1 minute ago
Showing 2 changed files with 2 additions and 2 deletions.
charts/jruedas-podinfo/Chart.yaml
charts/jruedas-podinfo/values.yaml
0 comments on commit 563004e

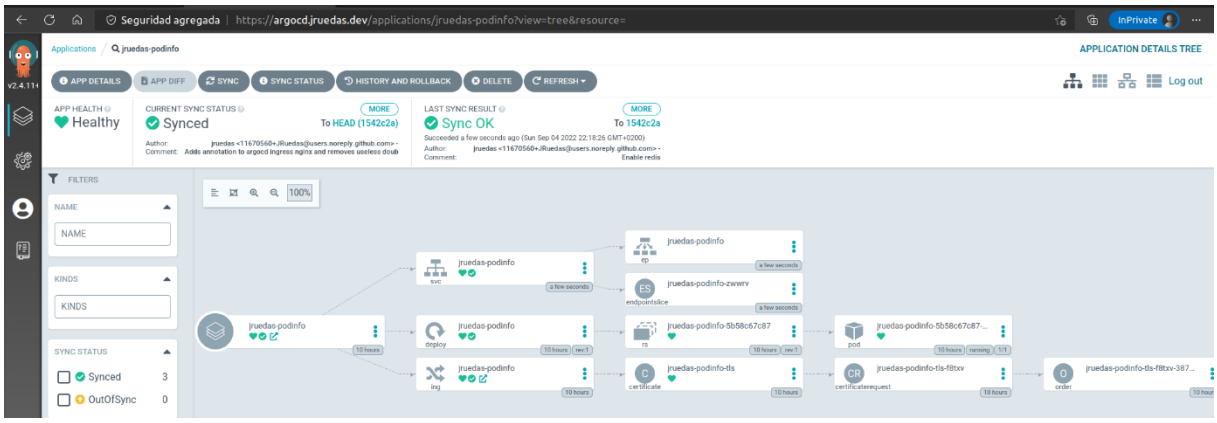

Seguridad agregada | https://argocd.jruedas.dev/applications/jruedas-podinfo?view=tree&resource=
APP HEALTH
Healthy
CURRENT SYNC STATUS
Synced
LAST SYNC RESULT
Sync OK
SHA: 563004e
Message: Version updated 1.0.14

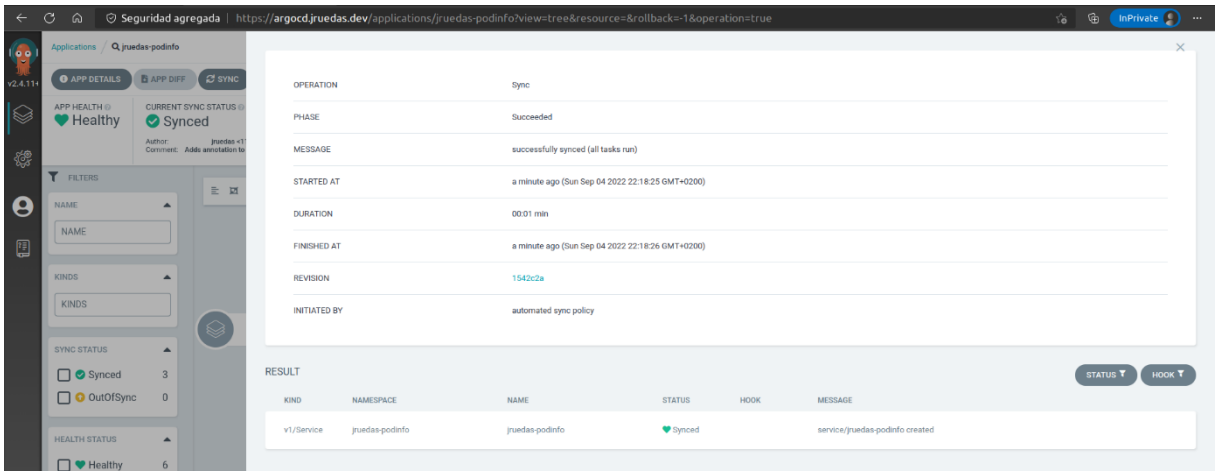

Seguridad agregada | https://podinfo.jruedas.dev
greetings from podinfo v1.0.14
Served by jruedas-podinfo-6579d4f8b8-l248t
This line has been commited to the source repo
PING
1.0.14
1.0.14

C-2. Ejemplo de corrección de deriva de configuración








Seguridad agregada | https://argocd.jruedas.dev/applications/jruedas-podinfo?view=tree&resource=&rollback=-1&operation=true
InPrivate
Applications / jruedas-podinfo
APP DETAILS
APP DIFF
SYNC
APP HEALTH
Healthy
CURRENT SYNC STATUS
Synced
FILTERS
NAME
KINDS
SYNC STATUS
Synced 3
OutOfSync 0
HEALTH STATUS
Healthy 6
OPERATION Sync
PHASE Succeeded
MESSAGE successfully synced (all tasks run)
STARTED AT a minute ago (Sun Sep 04 2022 22:18:25 GMT+0200)
DURATION 00:01 min
FINISHED AT a minute ago (Sun Sep 04 2022 22:18:26 GMT+0200)
REVISION 1542c2a
INITIATED BY automated sync policy
RESULT
STATUS
HOOK
KIND NAMESPACE NAME STATUS HOOK MESSAGE
v1/Service jruedas-podinfo jruedas-podinfo Synced service/jruedas-podinfo created