

GitOps

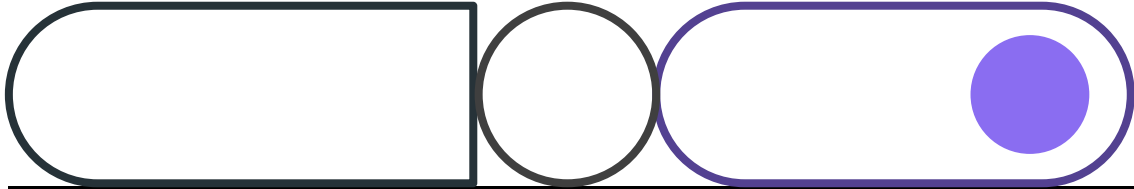
Índice

- 01 [Introducción](#)
- 02 [Herramientas GitOps](#)
- 03 [Evolución del DevOps](#)
- 04 [Administración de Infraestructura con GitOps](#)

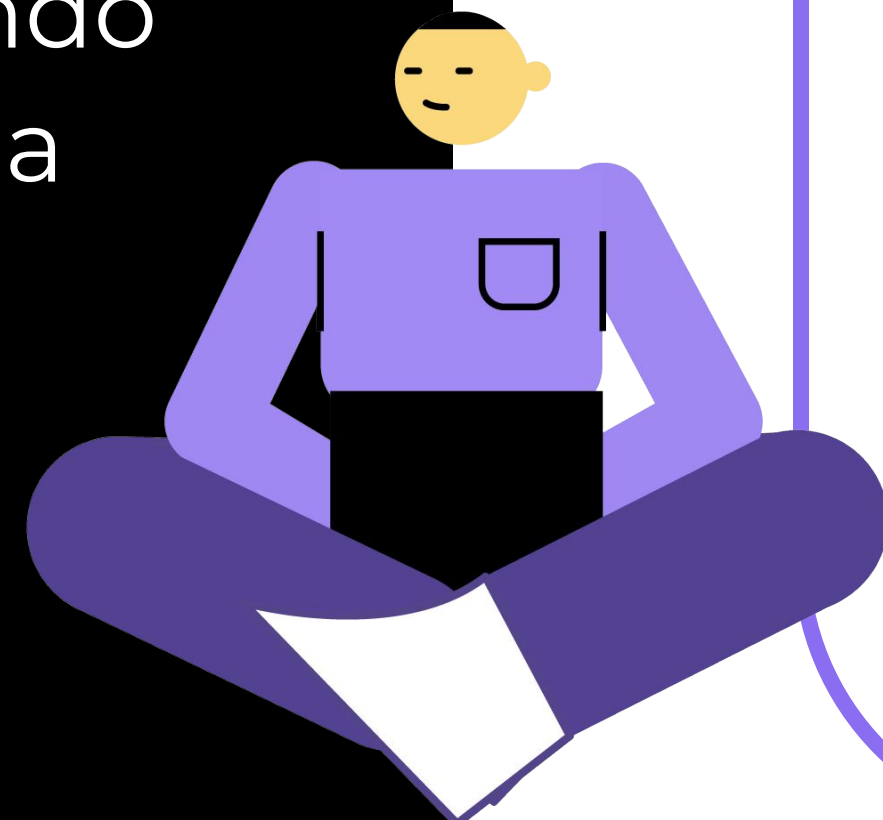


01

Introducción



GitOps consiste en una serie de prácticas para gestionar las configuraciones de las aplicaciones y las infraestructuras, usando Git como única fuente de verdad para la infraestructura declarativa y las aplicaciones.



02

Herramientas GitOps

Principios que debe seguir una herramienta GitOps

- El estado de un sistema o aplicación son siempre almacenados en Git.
- Git es siempre la fuente de verdad de lo que pasa en el sistema.
- Si se quiere cambiar el estado del sistema se debe ejecutar una operación Git; por ejemplo, crear un commit o abrir un pull request.
- Los procesos de despliegues, pruebas y rollbacks son controlados a través de un flujo Git.
- Una vez que el estado Git cambia, entonces el estado del entorno, plataforma o aplicativo debe coincidir con lo que está definido en el repositorio Git.
- Uso de agentes que se encargan de revisar los repositorios en busca de cambios o que serán informados de los mismos por mecanismos de aviso por parte del repositorio, y que controlarán si el sistema se encuentra en el estado especificado en Git.
- No se deberían realizar implementaciones manuales, cambios de plataforma, ni cambios de configuración de forma directa. Si un cambio es necesario, primero debe hacerse un commit en Git.

GitOps basado en agente y basado en push

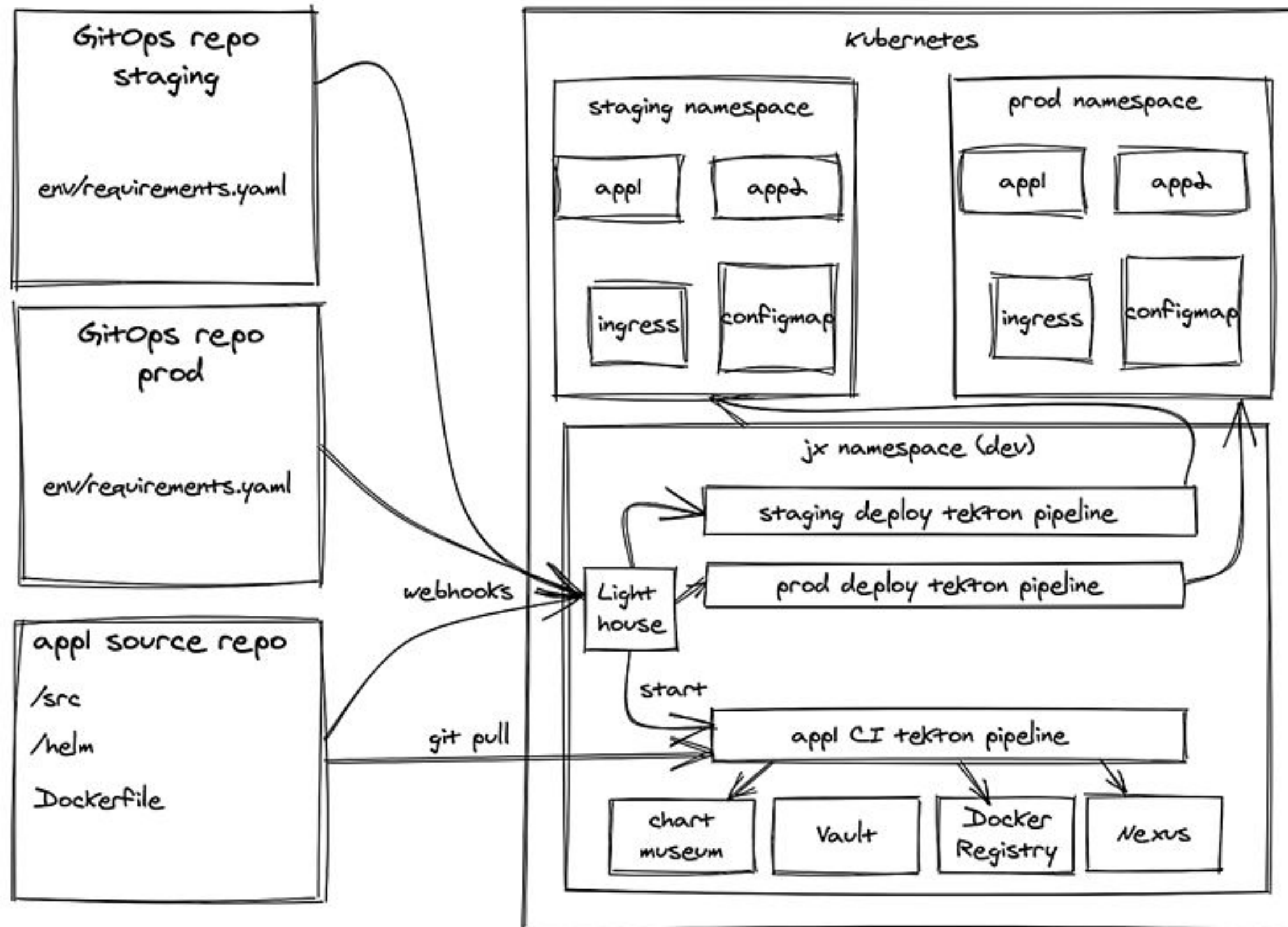
GitOps basado en agente

- **GitOps** basado en agente se refiere a ejecutar un proceso dentro de su infraestructura que facilita sus implementaciones.
- Un agente es un recurso que forma parte de su infraestructura. Se conectará periódicamente a su repositorio de Git y buscará cambios. Luego, el agente tomará medidas para aplicar los cambios obtenidos, lo que activará la transición de estado adecuada.
- Los agentes pueden proporcionar características adicionales como alertas, registros y monitoreo de implementación integrados. Estos lo mantienen constantemente informado de la actividad dentro de su infraestructura.

GitOps basado en push

- **GitOps** basado en push es el modelo por el cual la plataforma de Git o el proceso de CI avisan que existen cambios y/o ejecutan programas para realizar los cambios pertinentes.
- Debe configurar cada repositorio o flujo de CI para contar con la información adecuada, y así realizar los cambios en donde corresponde y asumir la responsabilidad de mantener sus scripts de implementación a lo largo del tiempo.
- Se puede utilizar herramientas ya conocidas y en las que se confía para el desarrollo, como kubectl o helm. Esto minimiza las diferencias entre las implementaciones locales y automáticas.

Jenkins X



Jenkins X es una solución de CI/CD. Sus capacidades permiten hacer uso de la metodología GitOps basado en push. Está desarrollado como un motor de CI/CD nativo de Kubernetes. Es de código abierto bajo licencia Apache 2.0 y está desarrollado por Cloudbees, que también ofrece una distribución comercial. Es importante tener en cuenta que, además de las capacidades de implementación basadas en GitOps, Jenkins X también cubre un espectro más amplio del ciclo de desarrollo, incluidas las fases de compilación y pruebas, típicas de un flujo de CI, como así también la compilación y el almacenamiento de imágenes de contenedores.

Capacidades de Jenkins X



Instalación

Se realiza por medio de un comando. El sistema se despliega en un cluster de K8S. Además, se aprovisionan el repositorio de configuración y los de estado de los entornos: desarrollo, staging y producción.



Despliegues con GitOps

Para desplegar un aplicativo compilado con el flujo de CI, es necesario actualizar el repositorio del entorno. Se puede configurar para que cuando se termine la compilación se actualice el repositorio de entorno de forma automática.



Administración

Las configuraciones propias de Jenkins X se administran por medio de un flujo de GitOps.



Multi-Tenant

Los usuarios y las aplicaciones comparten los mismos recursos, componentes y entornos, dado que no existe isolación entre equipos y aplicaciones en un cluster de Jenkins X.



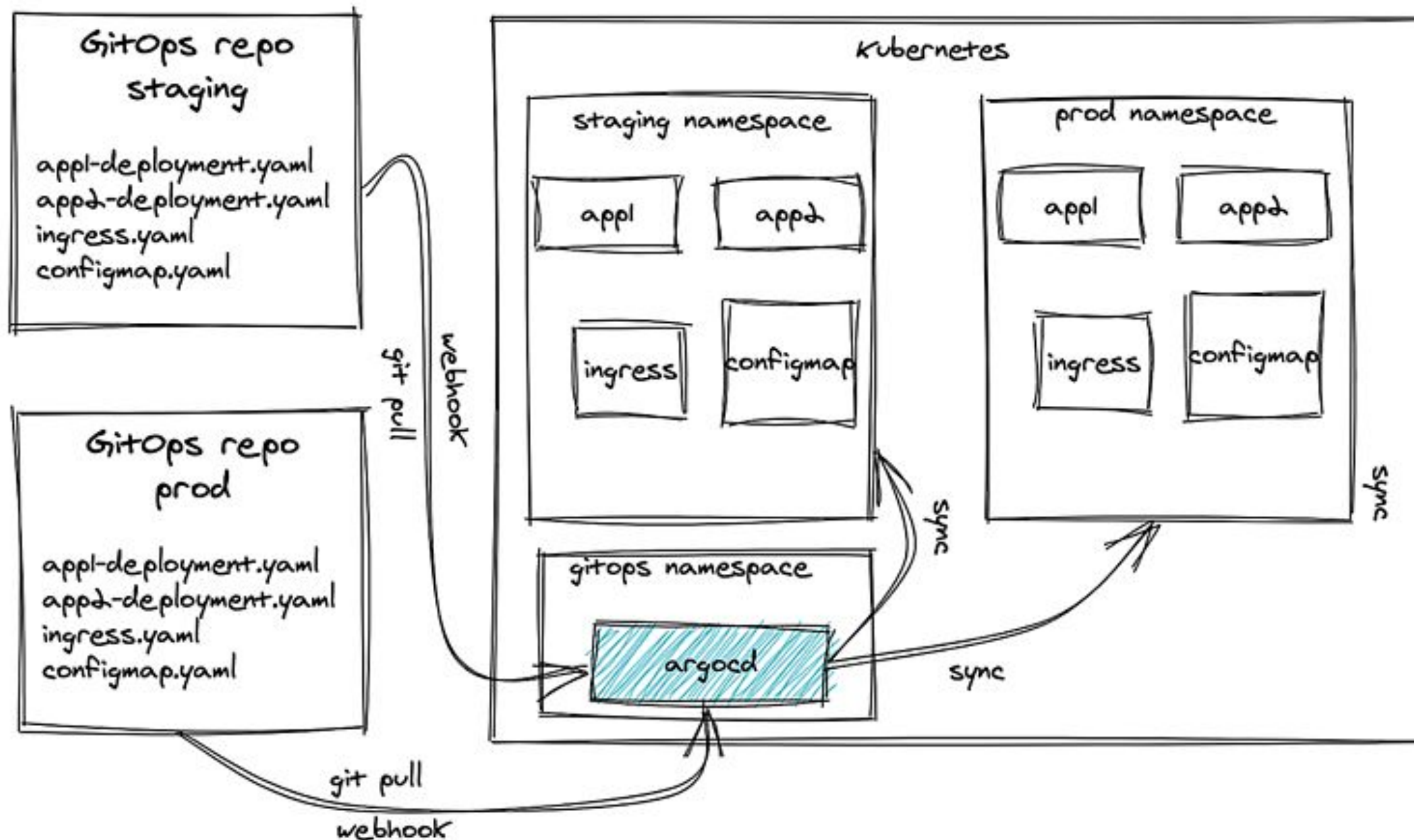
Multi-Cluster

Los entornos de las aplicaciones, como por ejemplo, staging o producción, pueden correr en otros clusters.

Características de Jenkins X

- ➔ Los proyectos se crean de forma simple y rápida. Con un comando se crea tanto el repositorio como un template básico de la aplicación y su helm chart. Además, se crea un pipeline básico de CI.
- ➔ Capacidad de poder crear repositorios directamente en Github.
- ➔ Se pueden crear pipelines de diversas tecnologías y metodologías.
- ➔ Uso de la metodología ChatOps (administración de procesos de operación por medio de mensajes de chat).
- ➔ Se puede previsualizar los cambios en un despliegue.

ArgoCD



ArgoCD es una herramienta de despliegue continuo (CD) basada en GitOps para Kubernetes. Se enfoca en la administración de despliegues de aplicaciones, pero consta de otras opciones como, por ejemplo, sincronización, controles de acceso de usuarios, verificaciones de estado, etc. Ha sido desarrollado desde 2018 por Intuit y es de código abierto bajo la licencia Apache 2.0.

Puede administrar varias aplicaciones, equipos y clusters.

Dispone de una interfaz web moderna y amigable en donde se pueden observar el estado de los aplicativos y administrar usuarios y permisos.

Capacidades de ArgoCD



Instalación

Se instala y gestiona de forma nativa en Kubernetes.



Despliegues con GitOps

ArgoCD monitorea un repositorio Git y aplica actualizaciones en los manifiestos del clúster si hay nuevos cambios en el repositorio. Se puede usar para desplegar aplicaciones o mantener configuraciones propias del cluster.



Administración

Como se ejecuta en su propio namespace y toda la configuración se almacena en ConfigMaps, Secrets y Custom Resources. Esto permite usar GitOps para administrar el propio ArgoCD. También dispone de una interfaz web.



Multi-Tenant

Una sola instancia de ArgoCD puede manejar muchas aplicaciones de diferentes equipos. Lo hace utilizando el concepto de Proyectos. El proyecto puede contener varias aplicaciones y se asignan a un equipo.



Multi-Cluster

ArgoCD puede sincronizar aplicaciones en el clúster de Kubernetes en el que se ejecuta y también puede administrar clústeres externos. Se puede configurar para que solo tenga acceso a un conjunto restringido de namespaces.

Características de ArgoCD

- Soporta varios formatos de manifiestos y de sistemas de gestión de despliegues en Kubernetes. Entre ellos, se destacan: Kustomize applications, Helm charts, Ksonnet applications, etc.
- Puede integrarse a herramientas de Configuration Management por medio de plugins.
- ArgoCD chequea si se produjeron cambios que no hayan sido realizados por el mismo, y en caso de detectar desvíos, realiza los ajustes necesarios para garantizar el estado definido en el repositorio de Git.
- Elimina recursos obsoletos durante el proceso de sincronización.

Conclusión

Tanto **ArgoCD** y **Jenkins X** tienen capacidades muy interesantes y poderosas para administrar implementaciones en Kubernetes, pero cada uno tiene ventajas y desventajas que deben evaluarse para su caso específico. Estas herramientas realmente no compiten entre sí, sino que apuntan a cumplir con diferentes casos de uso.

ArgoCD puede administrar implementaciones para múltiples aplicaciones en diferentes clústeres. Se ejecuta con permisos sobre en el clúster, pero también administra el acceso y los permisos para equipos y proyectos. Tiene una interfaz web elegante para administrar aplicaciones y proyectos, y ofrece un buen conjunto de características para administrar los despliegues.

Jenkins X es un conjunto de herramientas que sirven para crear un flujo de trabajo de desarrollo en torno a los repositorios de GitHub (pronto se admitirán otros proveedores). Ejecuta procesos de CI para compilar y realizar test de las aplicaciones, y además, ofrece gestión de eventos en función de los cambios en los repositorios de Git.

03

Evolución del DevOps

Desafíos de GitOps

Uno de los mayores desafíos a la hora de adoptar GitOps es la falta de madurez. El término sigue siendo abstracto para los equipos de operaciones que no tienen experiencia en los flujos de trabajo controlados por versiones.

Muchos equipos de infraestructura estarán acostumbrados a las prácticas de trabajo manual en las que han estado trabajando durante las últimas dos décadas. Generalmente, se conectan a los servidores por medio de SSH, realizan sus cambios y los documentan. No se mantiene un buen control de los cambios, pero es simple y funciona.

GitOps ayuda a mejorar la falta de control y puede brindar mejor visibilidad del estado de la plataforma y de los aplicativos. Cabe destacar que existe una curva de aprendizaje que implica aprender a trabajar utilizando un flujo de trabajo estructurado y adquirir conocimiento en un nuevo conjunto de herramientas muy diferentes a las utilizadas. Ya no sería necesario acceder por SSH a un sistema, sino que se realizarían cambios modificando archivos y esperando a que un proceso de CD los aplique.

Un desafío a comentar es la estrategia a utilizar cuando existen varios entornos diferentes. Un enfoque común es asignar a cada entorno su propia rama en el repositorio. En este caso, la gestión se complica si tiene muchos entornos. Un enfoque alternativo podría basarse en múltiples repositorios, quizás usando submódulos de Git, pero eso solo mueve la redundancia a otra parte.

GitOps es todavía un concepto joven y no existen modelos establecidos para su implementación. Sin una arquitectura de referencia, hay que buscar y probar con la estrategia que más se acomode.

DevOps y GitOps

DevOps, más allá de un rol, es también una filosofía en la cual hacer las cosas de la manera más automatizada e incorporando el conjunto de skills propios de Desarrollo y de Operaciones.

Implementar este conjunto de filosofías que conducen a automatizar las tareas de una compañía no es una tarea sencilla. Requiere coordinación entre los equipos y modificar cómo entendemos e interactuamos con el ciclo de vida de las aplicaciones (ALM). Actualmente, es natural pensar sobre una aplicación como un conjunto de piezas diferentes, orquestadas y trabajando armoniosamente en conjunto.

En los últimos años, hemos comenzado a crear software mucho más confiable, con metodologías ágiles y creando tests automáticos que evitan, en cierta forma, el error humano, beneficiando todo el ciclo del desarrollo de software con nuevas herramientas (Jenkins, CircleCI, Travis, Sonar, Linters) y nuevas metodologías (Testing automation e Integración Continua) pero, por su contrapartida, la infraestructura no tuvo el mismo nivel de evolución.

GitOps es una filosofía que está cambiando el paradigma y va a continuar haciéndolo en los próximos años. 10% es en herramientas, y un 90% en filosofía. Tener infraestructura como código (IaC) debe ser un objetivo para cualquier equipo de DevOps o SRE.

¿Qué aporta GitOps a DevOps?

- **Observabilidad:** los sistemas GitOps ofrecen monitoreo, registro, seguimiento y visualización en aplicaciones complejas para que los desarrolladores puedan ver qué está fallando y a dónde.
- **Control de versiones y gestión de cambios:** obviamente, este es un beneficio clave de usar un sistema de control de versiones como Git. Las actualizaciones defectuosas pueden ser fácilmente eliminadas/deshechas.
- **Fácil adopción:** GitOps se basa en las habilidades DevOps que muchos desarrolladores ya tienen.
- **Productividad:** GitOps proporciona mejoras en la productividad que DevOps y CI/CD han estado desarrollando.
- **Auditoría:** Gracias a Git, cada acción puede ser rastreada, facilitando el seguimiento de la causa de los errores.

04

Administración de Infraestructura con GitOps

Principios básicos de administración con GitOps

- Todo lo relacionado con la infraestructura se describe de forma declarativa. Todas las configuraciones de servidor se definen como definiciones y se describen en el repositorio de Git para lograr la máxima fiabilidad.
- Los estados del sistema están versionados en Git. Esta es una de las principales razones por las que recuperarse de un error con GitOps es muy fácil. Puede volver a una determinada versión de forma rápida y coherente.
- Los cambios deben aprobarse, y solo los aprobados pueden ser implementados. No es necesario un flujo de trabajo de implementación por separado ni un paquete previo de nuevos códigos antes de la implementación. Todo sucede automáticamente.
- La corrección y la seguridad se mantienen como partes del proceso. No es necesario incorporar flujos de trabajo de seguridad e integridad separados en los flujos de CI / CD.

¡Muchas gracias!