DOCUMENTO ARQUITECTURA DEVOPS ADA S.A.S. – PROYECTO MIGRACIÓN SICOF

Fábrica de software – ADA S.A.S.

**REVISIÓN Y CONTROL DE CAMBIOS**

**Revisión y Versiones**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Autor | Versión | Descripción | Cargo | Fecha |
| Jhonn Edison Gomez Melo | 1.0 | Documento | Desarrollador | 16/08/2024 |

**Control de Cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Autor | Versión | Descripción del cambio |
| 16/08/2024 | Jhonn Edison Gomez Melo | 1.0 | Liberación Documento |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

TABLA DE CONTENIDO

Contenido

[**RESUMEN** 4](#_Toc174541060)

[**INTRODUCCIÓN** 5](#_Toc174541061)

[**DIAGRAMA DE ARQUITECTURA DEVOPS** 6](#_Toc174541062)

[**COMMIT Y PUSH DE CÓDIGO EN GITLAB** 6](#_Toc174541063)

[**CONSTRUCCIÓN DEL PROYECTO EN JENKINS** 8](#_Toc174541064)

[**CONSTRUCCIÓN DE LA IMAGEN DOCKER** 9](#_Toc174541065)

[**PUBLICACIÓN DE LA IMAGEN DOCKER EN DOCKER HUB** 9](#_Toc174541066)

[**CREACIÓN DEL CONTENEDOR DOCKER EN KUBERNETES** 10](#_Toc174541067)

[**DESPLIEGUE EN LOS DIFERENTES ENTORNOS** 10](#_Toc174541068)

[**MONITOREO Y MANTENIMIENTO DE LA APLICACIÓN** 12](#_Toc174541069)

[**CONCLUSIÓNES** 13](#_Toc174541070)

# **RESUMEN**

La arquitectura DevOps en ADA S.A.S. está diseñada para automatizar el ciclo de vida completo del desarrollo de software, desde la codificación hasta la implementación y monitoreo en producción.

A través de la integración continua (CI) y el despliegue continuo (CD), la empresa ha logrado crear un flujo de trabajo que permite entregas rápidas y de alta calidad, manteniendo la flexibilidad necesaria para adaptarse a cambios y demandas del mercado.

Este documento describe el proceso en siete pasos principales, que van desde la codificación y el commit de código, hasta la creación, despliegue y monitoreo de contenedores en diferentes entornos. Cada paso utiliza tecnologías específicas que contribuyen a la automatización y mejora continua del proceso.

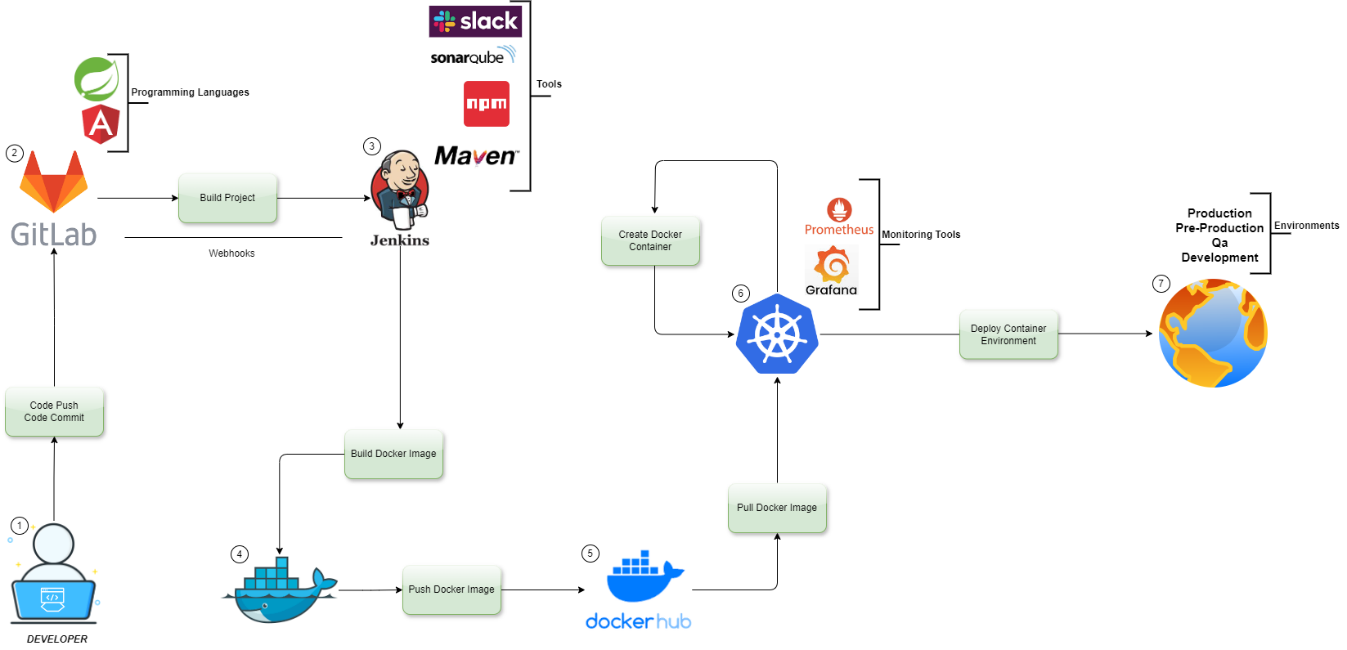
# **INTRODUCCIÓN**

La arquitectura DevOps ha revolucionado la manera en que las organizaciones desarrollan, prueban, implementan y monitorean aplicaciones. En ADA S.A.S., la implementación de esta metodología ha permitido mejorar significativamente la eficiencia y la calidad en la entrega de software, reduciendo el tiempo de lanzamiento al mercado y aumentando la satisfacción del cliente.

La integración de diversas tecnologías y herramientas en un flujo de trabajo coherente y automatizado es clave para mantener la agilidad y escalabilidad de los procesos de desarrollo y operaciones.

En este documento, se detalla la arquitectura DevOps implementada en ADA S.A.S., destacando cada uno de los pasos del proceso, las tecnologías utilizadas y cómo esta estructura se replica a través de todos los entornos (desarrollo, QA, preproducción y producción) para garantizar consistencia y eficiencia.

# **DIAGRAMA DE ARQUITECTURA DEVOPS**



Este documento se centra en proporcionar una descripción **funcional** de la arquitectura DevOps, destacando el flujo de trabajo y las herramientas empleadas en cada etapa del proceso de desarrollo y despliegue de aplicaciones. **Para obtener un enfoque más técnico, que incluya detalles sobre la configuración de la arquitectura y los pipelines específicos para cada rama del desarrollo, es necesario consultar el documento titulado "Manual y Especificación Técnica Creación De Pipelines".** El documento técnico proporciona instrucciones detalladas sobre la configuración de los pipelines, asegurando que se sigan las mejores prácticas en la implementación de la arquitectura DevOps.

## **COMMIT Y PUSH DE CÓDIGO EN GITLAB**



El ciclo de vida del desarrollo comienza cuando un desarrollador escribe y actualiza el código en su entorno local, utilizando tecnologías como Angular para el frontend y Spring Boot para el backend. Una vez completadas las modificaciones, el código se sube al repositorio GitLab mediante un commit y push, siguiendo el GitFlow definido por la compañía.

GitLab gestiona el control de versiones, permitiendo a los equipos trabajar de manera colaborativa, rastrear cambios y gestionar ramas de desarrollo. Al subir el código a la rama del repositorio específica (Develop, QA, Pre-Production, Production o máster), se desencadenan automáticamente los pipelines configurados en GitLab para la construcción y prueba del proyecto. Esto asegura que tanto el frontend como el backend sean compilados y validados de manera coherente antes de ser promovidos a los siguientes entornos.

**Tecnologías:**

* **GitLab:** Es una plataforma de DevOps completa que proporciona un flujo de trabajo centralizado para la gestión de código fuente, CI/CD, gestión de proyectos y más. GitLab facilita la colaboración entre equipos y la integración automatizada de código, asegurando un desarrollo más ágil y controlado.
* **Angular:** Es un framework para el desarrollo de aplicaciones web frontend, que permite construir interfaces de usuario modernas y responsivas. Angular se integra bien con otros servicios y APIs, lo que lo hace ideal para el desarrollo de aplicaciones de una sola página (SPA) en el entorno DevOps.

Un dibujo de una cara feliz

Descripción generada automáticamente con confianza baja

* **Spring Boot:** Es un framework que simplifica la creación de aplicaciones Java backend robustas y escalables. Spring Boot facilita la configuración y el despliegue de aplicaciones, permitiendo a los desarrolladores centrarse en la lógica empresarial en lugar de en la configuración de la infraestructura.

Icono

Descripción generada automáticamente

## **CONSTRUCCIÓN DEL PROYECTO EN JENKINS**

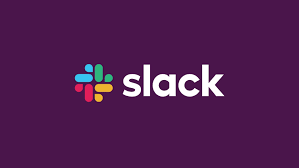
Un dibujo de un personaje animado

Descripción generada automáticamente con confianza media

Jenkins, a través de los webhooks configurados en GitLab, detecta automáticamente los cambios en el repositorio y ejecuta los pipelines definidos. En esta etapa, Jenkins compila el código, ejecuta pruebas unitarias y de integración, y asegura que el proyecto esté libre de errores antes de pasar a la siguiente fase. Esta fase es crucial para validar la integridad del código y asegurar que el producto es estable antes de avanzar.(***Para obtener más información sobre Jenkins, validar el documento Especificación Técnica Jenkins***)

**Tecnologías:**

**Jenkins:** Es una herramienta de automatización de código abierto que facilita la integración y el despliegue continuos. Jenkins orquesta y ejecuta pipelines, permitiendo la automatización de tareas como la construcción de proyectos, pruebas y despliegues. Además, se integra con diversas herramientas como SonarQube(***Para obtener más información sobre SonarQube, validar el documento Especificación Técnica SonarQube***) para análisis de código, Maven o Npm para la construcción de proyectos. Se utiliza Slack para notificaciones en tiempo real sobre el estado de las construcciones y ejecuciones de los pipelines. (***Para obtener más información sobre Slack, validar el documento Especificación Técnica Slack***)

 Texto, Logotipo

Descripción generada automáticamente Imagen que contiene dibujo

Descripción generada automáticamente Logotipo, nombre de la empresa

Descripción generada automáticamente

## **CONSTRUCCIÓN DE LA IMAGEN DOCKER**

Logotipo, nombre de la empresa

Descripción generada automáticamente con confianza media

Después de que el proyecto ha sido validado y aprobado en la fase de construcción, Jenkins procede a construir una imagen Docker. Docker encapsula la aplicación y sus dependencias en una imagen de contenedor, asegurando que la aplicación se ejecutará de manera consistente en cualquier entorno. Esta imagen es el artefacto que se promoverá a través de los diferentes entornos (desarrollo, QA, pre-producción, producción).

**Tecnologías:**

**Docker:** Docker es una plataforma de contenedores que permite a los desarrolladores empaquetar aplicaciones junto con todas sus dependencias en una imagen estándar. Estas imágenes se pueden ejecutar de manera consistente en cualquier entorno que soporte Docker, lo que facilita la portabilidad y la escalabilidad de las aplicaciones.(***Para obtener más información sobre Docker, validar el documento Especificación Técnica Docker***)

## **PUBLICACIÓN DE LA IMAGEN DOCKER EN DOCKER HUB**

Imagen que contiene Logotipo

Descripción generada automáticamente

Una vez creada la imagen Docker, esta se envía a un registro centralizado, en este caso, Docker Hub. Al publicar la imagen en Docker Hub, se asegura que esté disponible para ser desplegada en cualquier entorno de la organización. Esto facilita la distribución de la aplicación y asegura que todas las instancias de la aplicación utilicen la misma imagen.

**Tecnologías:**

**Docker Hub:** Es un servicio de registro de imágenes Docker en la nube. Docker Hub almacena, distribuye y gestiona imágenes Docker, proporcionando un lugar centralizado desde el cual las imágenes pueden ser descargadas y utilizadas en diferentes entornos de despliegue. (***Para obtener más información sobre Docker Hub, validar el documento Especificación Técnica Docker Hub***)

## **CREACIÓN DEL CONTENEDOR DOCKER EN KUBERNETES**

Icono

Descripción generada automáticamente

Kubernetes, el sistema de orquestación de contenedores se encarga de descargar la imagen desde Docker Hub y crear un contenedor en el entorno especificado. Kubernetes maneja la distribución y ejecución de contenedores, asegurando que la aplicación esté correctamente balanceada y escalada según las necesidades del entorno. Esta fase es clave para la gestión de aplicaciones en todos los entornos de despliegue, ya que asegura la alta disponibilidad y escalabilidad.

**Tecnologías:**

**Kubernetes:** Kubernetes es un sistema de orquestación de contenedores de código abierto que automatiza la implementación, escalado y gestión de aplicaciones en contenedores. Facilita la creación de entornos de despliegue flexibles y escalables, asegurando que las aplicaciones puedan manejar cargas de trabajo variables sin interrupciones. (***Para obtener más información sobre Docker, validar el documento Especificación Técnica Kubernetes***)

## **DESPLIEGUE EN LOS DIFERENTES ENTORNOS**

Con la imagen ya desplegada en un contenedor de Kubernetes, esta se implementa en los diferentes entornos: desarrollo, QA, pre-producción o producción. El despliegue en cada entorno sigue el mismo proceso automatizado, lo que garantiza consistencia y elimina la posibilidad de errores humanos. Además, este despliegue será automatizado a través de pipelines CI/CD, lo que acelera el proceso y permite entregas continuas.

**Tecnologías:**

Herramientas CI/CD: Jenkins, junto con otras herramientas CI/CD, automatizan el proceso de despliegue, garantizando que las aplicaciones se implementen de manera consistente y eficiente en todos los entornos.

**Continuous Integration (CI):** Es una práctica de desarrollo de software en la que los desarrolladores integran su código en un repositorio compartido de manera frecuente, a menudo varias veces al día. Cada integración es verificada por un proceso automatizado que incluye la construcción del código y la ejecución de pruebas. El objetivo principal de CI es identificar errores lo antes posible, lo que facilita su corrección y reduce el riesgo de que los problemas se acumulen.

* **Objetivo:** Detectar y solucionar errores rápidamente, mejorar la colaboración entre desarrolladores y mantener un código base que siempre esté en un estado que pueda ser desplegado.

**Continuous Delivery (CD):** Es una práctica que se extiende más allá de CI. Se refiere a la capacidad de lanzar automáticamente nuevas versiones de software que han pasado todas las pruebas desde un entorno de desarrollo a un entorno de pre-producción o producción. Aunque el despliegue en producción puede ser manual, la entrega del software a un entorno listo para producción es completamente automatizada.

* **Objetivo:** Asegurar que el software esté siempre en un estado listo para ser desplegado en cualquier momento, minimizando el riesgo y el tiempo necesario para realizar un lanzamiento.

**Continuous Deployment (CD):** Es el siguiente paso después de Continuous Delivery. En esta práctica, cada cambio que pasa las pruebas automatizadas es desplegado automáticamente, sin intervención manual. Esto permite que los cambios lleguen a los usuarios finales lo más rápido posible.

* **Objetivo:** Automatizar completamente el proceso de despliegue, permitiendo que cada cambio en el código que pase las pruebas llegue inmediatamente a producción.

**Diferencias Clave:**

* **CI (Integración Continua):** Enfocado en la integración de código y verificación mediante pruebas automáticas.
* **CD (Entrega Continua):** Asegura que el software esté en un estado listo para ser desplegado en cualquier momento.
* **CD (Despliegue Continuo):** Automatiza el despliegue en producción de manera que cualquier cambio aprobado por las pruebas automáticas se libera automáticamente a los usuarios.

## **MONITOREO Y MANTENIMIENTO DE LA APLICACIÓN**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteLogotipo

Descripción generada automáticamente

Una vez desplegada la aplicación, su monitoreo es fundamental para asegurar su correcto funcionamiento. Prometheus se encarga de recolectar métricas en tiempo real, mientras que Grafana proporciona paneles visuales para el análisis de estas métricas. Estas herramientas permiten detectar problemas potenciales de manera proactiva y asegurar que la aplicación cumpla con los niveles de servicio esperados. El monitoreo constante también facilita la toma de decisiones informadas para mejoras futuras.

**Tecnologías:**

* **Prometheus:** Es un sistema de monitoreo y alerta de código abierto que recopila métricas de los sistemas y servicios en tiempo real. Prometheus ofrece un lenguaje de consulta potente para analizar datos y crear alertas basadas en esas métricas.
* **Grafana:** Es una plataforma de análisis que permite visualizar las métricas recopiladas por Prometheus y otras fuentes de datos, proporcionando paneles interactivos que facilitan la monitorización en tiempo real y el análisis histórico de la operación de los sistemas.

# **CONCLUSIÓNES**

Esta arquitectura DevOps implementada en ADA S.A.S. es esencial para asegurar un ciclo de vida de desarrollo y operaciones eficiente, automatizado y escalable. A través de la integración continua (CI) y el despliegue continuo (CD), se logra reducir el tiempo de entrega de nuevas funcionalidades y la detección temprana de errores, lo que mejora la calidad del software y la satisfacción del cliente.

Además, el uso de imágenes, contenedores y la orquestación con Kubernetes aseguran que las aplicaciones sean altamente portables y escalables, mientras que las herramientas de monitoreo proporcionan una visibilidad completa y continua de la operación de los sistemas.

Es importante destacar que esta arquitectura DevOps no se limita a un único entorno, sino que se replica de manera consistente en todos los entornos, ya sea desarrollo, QA, pre-producción o producción. Esto garantiza que los procesos, herramientas y prácticas sean uniformes a lo largo de todo el ciclo de vida del software, lo que facilita la transición entre ambientes y minimiza los riesgos asociados con las diferencias de configuración y operación entre ellos. La replicación de esta arquitectura en todos los entornos asegura que los equipos puedan desarrollar, probar y desplegar con confianza, sabiendo que los entornos son equivalentes y que los mismos procesos automatizados están en funcionamiento en cada etapa.