DOCUMENTO ESPECIFICACIÓN TÉCNICA DOCKER – 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](#_Toc174542007)

[**INTRODUCCIÓN** 5](#_Toc174542008)

[**¿QUÉ ES DOCKER?** 6](#_Toc174542009)

[**¿PARA QUÉ SIRVE DOCKER?** 6](#_Toc174542010)

[**REQUISITOS DEL SISTEMA:** 7](#_Toc174542011)

[**CREACION DE VOLUMEN DE JENKINS** 8](#_Toc174542012)

[**¿QUÉ SON LOS VOLÚMENES EN DOCKER?** 8](#_Toc174542013)

[**INSTRUCCIONES DE INSTALACIÓN DE LA IMAGEN DOCKERIZADA DE JENKINS** 10](#_Toc174542014)

[**INSTALACIÓN DEL CONTENEDOR DOCKERIZADO DE JENKINS** 11](#_Toc174542015)

[**CONCLUSIÓN** 13](#_Toc174542016)

# **RESUMEN**

Este documento explora Docker, una tecnología esencial en el ámbito de la virtualización y la gestión de contenedores. Comienza explicando qué es Docker y para qué se utiliza, destacando su papel en el desarrollo y despliegue de aplicaciones. Luego, se detallan los requisitos del sistema necesarios para ejecutar Docker de manera efectiva. A continuación, se aborda la creación de volúmenes en Jenkins, explicando también qué son los volúmenes en Docker y cómo se utilizan para gestionar el almacenamiento persistente en contenedores. Finalmente, se incluyen instrucciones claras para la instalación de imágenes Docker, proporcionando una guía práctica para los usuarios que deseen comenzar a utilizar Docker en sus proyectos.

# **INTRODUCCIÓN**

Docker ha revolucionado la forma en que las aplicaciones se desarrollan, despliegan y ejecutan, permitiendo a los desarrolladores empaquetar software en contenedores que incluyen todo lo necesario para funcionar en cualquier entorno.

Este documento tiene como objetivo proporcionar una comprensión clara de Docker, comenzando con una definición de qué es Docker y para qué se utiliza. A continuación, se detallarán los requisitos del sistema necesarios para su correcta instalación y funcionamiento.

Se abordará también el concepto de volúmenes en Docker y cómo estos se integran con Jenkins para gestionar el almacenamiento persistente. Finalmente, se ofrecerán instrucciones detalladas para la instalación de imágenes Docker, facilitando así el inicio del uso de esta potente herramienta en entornos de desarrollo y producción.

## **INTRODUCCIÓN A LAS URLS O VARIABLES**

En este documento, todas las URLs o variables utilizadas están almacenadas en la Wiki, la se encuentra en la siguiente ruta:

**Ruta de variables de Configuración:**

[**URL\_WIKI**/doku.php?id=ada:howto:sicoferp:factory:new-migracion-sicoferp:rutas\_de\_conexiones](http://10.1.20.89/doku.php?id=ada:howto:sicoferp:factory:new-migracion-sicoferp:rutas_de_conexiones)

A continuación, se listan las variables utilizadas en este documento para referirse a las URLs o variables, cabe resaltar que estas pueden cambiar dependiendo del ambiente en el que se está utilizando el enfoque:

* **URL\_SONAR**: URL para el servidor de SonarQube, ambiente de Desarrollo.
* **URL\_WIKI:** URL del repositorio Wiki

# **¿QUÉ ES DOCKER?**



Docker es una plataforma de código abierto diseñada para permitir la creación, el despliegue y la ejecución de aplicaciones dentro de contenedores. Los contenedores son unidades de software que encapsulan todas las dependencias necesarias para ejecutar una aplicación, incluidos el código, las bibliotecas y las configuraciones del sistema. Docker proporciona una manera eficiente y portátil de empaquetar, distribuir y ejecutar aplicaciones, independientemente del entorno de ejecución. Utiliza la tecnología de contenedores para permitir la creación de entornos de ejecución aislados y ligeros, conocidos como contenedores. Cada contenedor ejecuta una aplicación o un servicio de manera independiente de otros contenedores y del sistema operativo subyacente.

# **¿PARA QUÉ SIRVE DOCKER?**

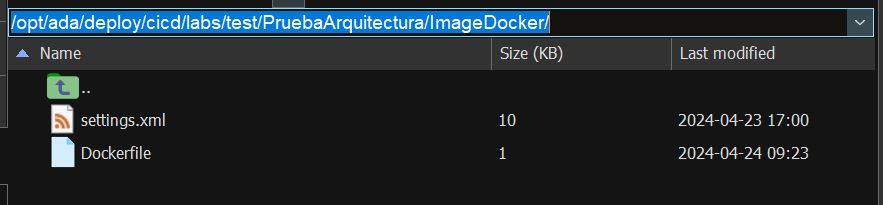
Docker sirve para varios propósitos relacionados con el desarrollo, el despliegue y la gestión de aplicaciones:

* Desarrollo de aplicaciones: Docker facilita la creación de entornos de desarrollo reproducibles y portátiles, permitiendo a los desarrolladores trabajar en aplicaciones con las mismas configuraciones y dependencias, independientemente de su entorno de desarrollo.
* Despliegue de aplicaciones: Docker simplifica el proceso de despliegue de aplicaciones al empaquetarlas junto con todas sus dependencias en contenedores. Esto facilita la implementación de aplicaciones en cualquier entorno, ya sea local, en la nube o en servidores dedicados.
* Gestión de infraestructura: Docker proporciona herramientas para gestionar y orquestar contenedores en entornos de producción. Esto incluye la capacidad de escalar automáticamente los contenedores, administrar el ciclo de vida de las aplicaciones y automatizar tareas de administración.
* Entrega continua y DevOps: Docker se integra fácilmente en prácticas de entrega continua y en metodologías de desarrollo de operaciones (DevOps), permitiendo a los equipos de desarrollo automatizar el proceso de construcción, prueba y despliegue de aplicaciones de manera eficiente y repetible.
* Microservicios: Docker es una tecnología clave en la implementación de arquitecturas de microservicios, donde las aplicaciones se dividen en componentes independientes y desacoplados, cada uno ejecutándose en su propio contenedor.

Se realiza el proceso de instalación de la imagen dockerizada de Jenkins, con el fin de poder crear un contenedor a partir de dicha imagen y de esa manera poder ejecutar Jenkins en el servidor especificado (Para este caso el servidor de desarrollo). Para dicha instalación se definieron los siguientes parámetros:

# **REQUISITOS DEL SISTEMA:**

En el servidor de Desarrollo (**URL\_SONAR**) se realiza la instalación del Dockerfile, el cual estará encargado de generar la imagen de Jenkins adaptándose a las dependencias de software que tiene la compañía en los repositorios Nexus, las cuales se configuran en un archivo settings.xml y se ubican ambos archivos en la siguiente ruta /opt/ada/deploy/cicd/labs/test/PruebaArquitectura/ImageDocker/



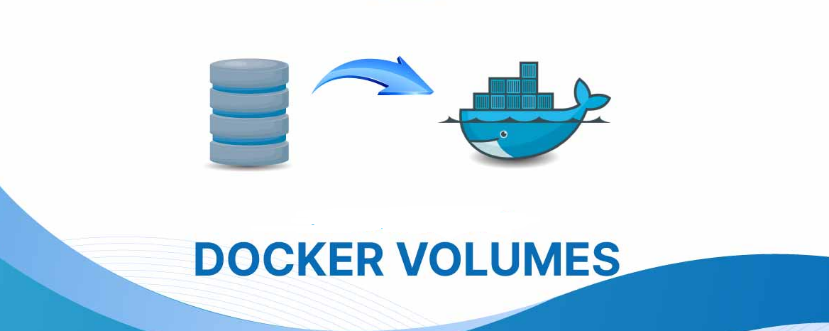
En el archivo Dockerfile se pueden evidenciar los requisitos con los cuales se instalará Jenkins, específicamente la instalación de la imagen tiene estas características:

* Utiliza la imagen base de Jenkins con JDK 21.
* Cambia al usuario root para poder instalar paquetes y herramientas adicionales.
* Define las versiones de Maven y kubectl que se instalarán.
* Actualiza los paquetes del sistema e instala herramientas necesarias como wget, apt-transport-https, ca-certificates, curl, gnupg-agent, y software-properties-common.
* Agrega el repositorio de Docker y su clave GPG, luego instala Docker.
* Agrega el usuario jenkins al grupo docker para permitirle ejecutar comandos Docker.
* Descarga y extrae Maven, configura enlaces simbólicos para facilitar su uso, y establece los permisos adecuados.
* Descarga e instala kubectl.
* Establece la variable de entorno MAVEN\_HOME.
* Cambia de nuevo al usuario Jenkins para ejecutar el contenedor con menos privilegios.
* Copia un archivo settings.xml personalizado a la configuración de Maven del usuario Jenkins.

Este Dockerfile configura una imagen de Jenkins que incluye todas las herramientas necesarias para la construcción de proyectos con Maven y el despliegue de aplicaciones en Kubernetes, además de estar lista para ejecutar comandos Docker.

# **CREACION DE VOLUMEN DE JENKINS**

## **¿QUÉ SON LOS VOLÚMENES EN DOCKER?**



Los volúmenes en Docker son una forma de persistir datos generados y utilizados por contenedores Docker. Los volúmenes se almacenan en el sistema de archivos del host fuera de los directorios de copias en capas de contenedores, y son gestionados por Docker. Los volúmenes son la forma preferida para mantener datos que deben persistir más allá del ciclo de vida de un contenedor, por varias razones:

* Persistencia: Los datos almacenados en volúmenes persisten incluso si el contenedor es eliminado.
* Compartición de Datos: Los volúmenes permiten compartir datos entre contenedores.
* Rendimiento: En la mayoría de los casos, los volúmenes proporcionan un mejor rendimiento en comparación con la unión de archivos del sistema de contenedores.
* Backups y Restores: Es fácil realizar copias de seguridad y restaurar datos usando volúmenes.



Descripción del comando: Este comando ejecuta un contenedor temporal de busybox que copia los datos de Jenkins desde otro contenedor a un directorio de backup en el host.

**Desglose del comando**:

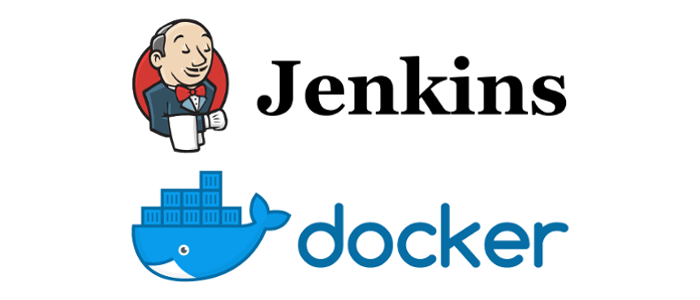
* docker run: Inicia un nuevo contenedor.
* --rm: Elimina el contenedor automáticamente después de que se detiene.
* --volumes-from ecosystem-jenkins: Monta los volúmenes desde el contenedor ecosystem-jenkins en el nuevo contenedor. Esto significa que los datos y configuraciones en los volúmenes de ecosystem-jenkins estarán disponibles en el nuevo contenedor.
* -v /home/jenkins/volumes:/backup: Monta el directorio /home/jenkins/volumes del host en el contenedor como /backup. Esto permite que los datos copiados estén disponibles en el host.
* busybox: Usa la imagen de busybox, una pequeña imagen de Linux que es útil para tareas básicas y scripts.
* sh -c "cp -r /var/jenkins\_home /backup/": Ejecuta el comando sh dentro del contenedor que copia recursivamente (cp -r) los datos de /var/jenkins\_home a /backup.

**Función del comando:**

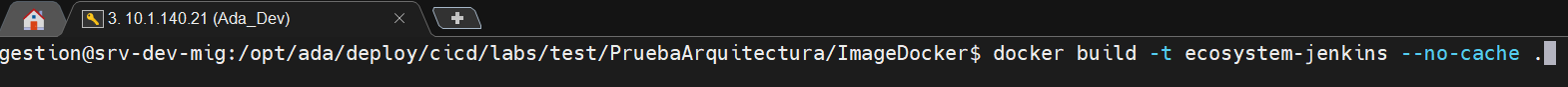
* Copiar Datos: El comando copia todo el contenido del directorio /var/jenkins\_home (que contiene los datos de Jenkins) del contenedor ecosystem-jenkins al directorio /home/jenkins/volumes en el host.
* Backup: Este procedimiento es útil para crear una copia de seguridad de los datos de Jenkins. Los datos copiados a /home/jenkins/volumes en el host pueden ser utilizados para restaurar o migrar Jenkins a otro entorno.

Este enfoque es útil para realizar copias de seguridad regulares de los datos importantes de Jenkins sin detener el contenedor ecosystem-jenkins.

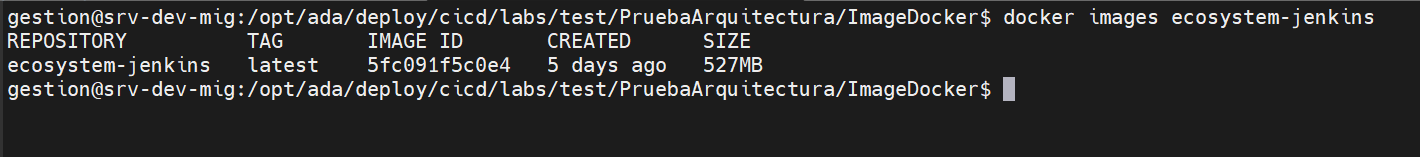
# **INSTRUCCIONES DE INSTALACIÓN DE LA IMAGEN DOCKERIZADA DE JENKINS**



Para realizar la instalación de la imagen dockerizada de Jenkins, debemos ingresar al servidor y acceder por consola a la siguiente ruta, /opt/ada/deploy/cicd/labs/test/PruebaArquitectura/ImageDocker/ en la cual se encuentran los archivos Dockerfile y settings.xml. Donde se encuentra toda la configuración para la instalación, una vez allí se debe ejecutar el siguiente comando para la creación:



Una vez creada la imagen ejecutamos el siguiente comando para realizar la validación de que la imagen jenkins dockerizada fue creada correctamente



Al ejecutar el comando, Docker nos muestra la imagen que tiene desplegada con el nombre que le dimos al crearla para este caso ecosystem-jenkins.

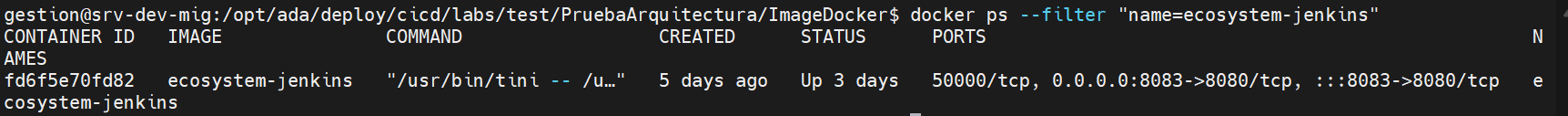
# **INSTALACIÓN DEL CONTENEDOR DOCKERIZADO DE JENKINS**

Una vez creada la imagen dockerizada de Jenkins, procedemos a realizar la creación del contenedor, con el siguiente comando:

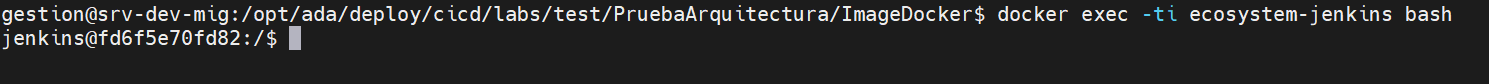


Aquí se especifica el puerto en el cual Jenkins va a ser desplegado (en este caso, 8083). Además, se le asigna un nombre al contenedor (ecosystem-jenkins) y se asegura que se reinicie automáticamente si se detiene (--restart always). Se monta el directorio /home/jenkins/volumes/jenkins\_home del host en el contenedor como /var/jenkins\_home para persistir los datos de Jenkins, y se monta el socket Docker del host (/var/run/docker.sock) en el contenedor para permitir que Jenkins controle Docker.

Al ejecutar el comando se genera el contenedor, para estar seguros de que se desplegó el contenedor correctamente ejecutamos el siguiente comando por consola:



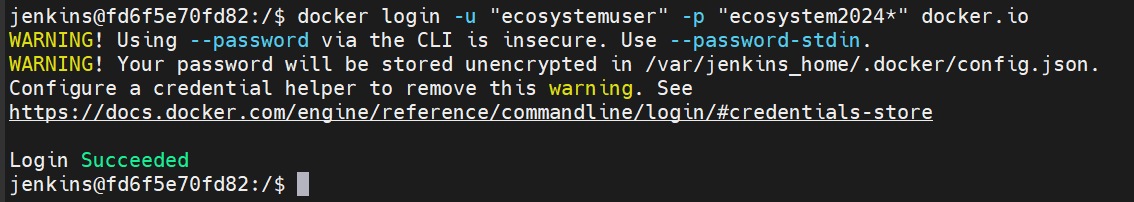
Una vez arriba procedemos a ingresar al contenedor para Validar las versiones de la instalación e instalar docker en el mismo, para ingresar al contenedor utilizamos el siguiente comando:



Dentro del contenedor procedemos a verificar la instalación de docker dentro del contenedor Jenkins ejecutando el siguiente comando:

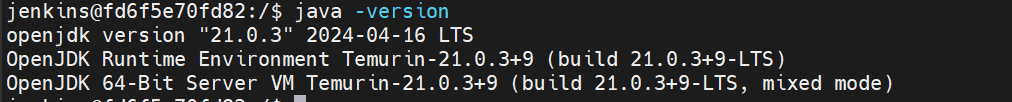


Para sincronizar el contenedor con el repositorio de Docker Hub, se debe ejecutar el siguiente comando:



Aquí se especifican las credenciales de docker hub con el que el contenedor tendrá comunicación para poder publicar las imágenes desde Jenkins hacia el registry de docker.

Por último para Validar la instalación ejecutamos el comando java -version para ver que el jdk haya quedado correctamente instalado:



# **CONCLUSIÓN**

Este documento ha ofrecido una exploración detallada de Docker, una herramienta clave en la virtualización de aplicaciones y la gestión de contenedores. Comenzamos definiendo qué es Docker y para qué se utiliza, destacando su capacidad para facilitar el desarrollo, despliegue y operación de aplicaciones en cualquier entorno. Los requisitos del sistema fueron detallados para asegurar que Docker funcione de manera óptima.

Se abordó la creación de volúmenes en Jenkins, explicando la importancia de los volúmenes en Docker para la gestión del almacenamiento persistente en contenedores. Estos volúmenes permiten que los datos persistan entre ejecuciones de contenedores, lo que es crucial para aplicaciones que manejan datos críticos.

Finalmente, se proporcionaron instrucciones claras y prácticas para la instalación de imágenes Docker y la configuración de contenedores Dockerizados de Jenkins, permitiendo a los usuarios implementar y gestionar sus entornos de desarrollo y producción de manera eficiente.

En resumen, este documento ha equipado a los lectores con los conocimientos necesarios para comprender y utilizar Docker de manera efectiva, desde la teoría hasta la práctica, facilitando su integración en procesos de desarrollo moderno. Con esta guía, los usuarios están mejor preparados para aprovechar las ventajas de Docker y Jenkins en sus proyectos, optimizando la forma en que desarrollan, prueban y despliegan aplicaciones.