UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS COMPUTACIONALES

TEMA:

ESTUDIO DEL CONTENEDOR CLOUD DOCKER Y PROPUESTA DE IMPLEMENTACIÓN PARA LA PLATAFORMA CLOUD FICA

APLICATIVO:

SISTEMA DE JUEGOS DEPORTIVOS

AUTOR:

MILTON ANDRÉS SIMBAÑA ALARCÓN

DIRECTOR:

ING. XAVIER MAURICIO REA PEÑAFIEL, MSC.

Ibarra-Ecuador
2016

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto de Repositorio Digital Institucional, determina la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto y pongo a disposición la siguiente información:

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD	1003862792
APELLIDOS Y NOMBRES	Milton Andrés Simbaña Alarcón
DIRECCIÓN	Ibarra, Obando luna y Maldonado.
EMAIL	masimbaniaa@utn.edu.ec
TELÉFONO FIJO	2951-513
TELÉFONO MOVIL	0991011494

DATOS DE LA OBRA	
TÍTULO	ESTUDIO DEL CONTENEDOR CLOUD DOCKER Y PROPUESTA DE IMPLEMENTACIÓN PARA LA PLATAFORMA CLOUD FICA
AUTOR	Milton Andrés Simbaña Alarcón
FECHA	24 de noviembre del 2016
PROGRAMA	Pregrado
TÍTULO POR EL QUE OPTA	Ingeniería en Sistemas Computacionales
DIRECTOR	Ing. Xavier Mauricio Rea Peñafiel, Msc.

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Milton Andrés Simbaña Alarcón, con cédula de identidad Nro. 100386279-2, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y el uso del archivo digital en la biblioteca de la universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión, en concordancia con la Ley de Educación Superior Artículo 144.

3. CONSTANCIA

El autor manifiesta que la obra objeto de la presente autorización es original y se desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Firma
Milton Andrés Simbaña Alarcón
1003862792

Ibarra, 24 de noviembre del 2016

UNIVERSIDAD TÉCNICA DEL NORTE

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE INVESTIGACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Milton Andrés Simbaña Alarcón, con cédula de identidad Nro. 100386279-2, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de propiedad intelectual del Ecuador, artículo 4, 5 y 6, en calidad de autor del trabajo denominado: ESTUDIO DEL CONTENEDOR CLOUD DOCKER Y PROPUESTA DE IMPLEMENTACIÓN PARA LA PLATAFORMA CLOUD FICA, que ha sido desarrollado para optar el título de Ingeniería en Sistemas Computacionales, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra mencionada, aclarando que el trabajo aquí descrito es de mi autoría y que no ha sido previamente presentado para ningún grado o calificación profesional.

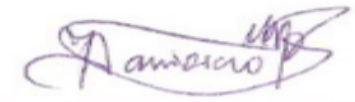
En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la biblioteca de la Universidad Técnica del Norte.

Firma
Milton Andrés Simbaña Alarcón
1003862792

Ibarra, 24 de noviembre del 2016

CERTIFICACIÓN

Certifico que el proyecto de Trabajo de Grado ESTUDIO DEL CONTENEDOR CLOUD DOCKER Y PROPUESTA DE IMPLEMENTACIÓN PARA LA PLATAFORMA CLOUD FICA. Ha sido realizado en su totalidad por el señor Milton Andrés Simbaña Alarcón portador de la cédula de identidad número: 100386279-2.



Ing. Xavier Mauricio Rea Peñafiel, Msc.

DIRECTOR DE PROYECTO

DEDICATORIA

El presente proyecto de titulación va dedicado a Dios por haberme brindado la vida, y la oportunidad de permitirme estudiar en tan emblemática institución, además de siempre situar a las personas indicadas en el lapso de mi vida.

A la universidad técnica del norte por haberme permitido crecer día a día como profesional brindándome docentes de calidad.

A mi amor Paulina Jácome por haberme apoyado siempre en momentos buenos y malos y además por ayudarme a culminar esta meta trazada.

A mis padres Milton Simbaña y Ruby Alarcón quienes con su ejemplo diario de lucha han reflejado en mí, sus buenos valores, por haberme guiado siempre por el camino correcto y finalmente por permitirme seguir estudiando y obtener esta carrera.

A mis hermanos por el apoyo moral y sentimental que me han brindado en el desarrollo de este proyecto.

A mis Docentes por haberme guiado en el desarrollo de este proyecto y además por haber compartido sus conocimientos en las aulas a lo largo de mi vida estudiantil.

Milton Andrés Simbaña Alarcón

AGRADECIMIENTOS

Primero que todo quiero agradecer a Dios al darme la oportunidad de cumplir esta meta, ya que jamás me abandonó y me ayudo a confiar en mí mismo y seguir adelante día a día.

Quiero agradecer también a mis Padres por apoyarme y confiar mí, gracias por ser los mejores padres, por tolerarme, por consentirme y nunca dejar de amarme.

A mis hermanos y sobrinos por entenderme y brindarme su ayuda siempre en momentos que más lo he necesitado.

Agradezco también a mi novia Paulina Jácome quién ha sido mi compañera desde el inicio de mi carrera, quien me ayudó a seguir a delante a no rendirme y hacerme crecer como persona, gracias por la paciencia y el tiempo que compartiste conmigo a lo largo de toda nuestra carrera.

Gracias al Ing. Xavier Mauricio Rea Peñafiel, Msc. Quién ha sido un amigo en las aulas y un excelente tutor en el desarrollo de mi trabajo final.

Contenidos

Índice de tablas

Índice de figuras

Resumen

Este proyecto se realizó utilizando conceptos basados en la nube y de programación en el cual se efectuó un estudio de contenedores Cloud con el fin de administrar aplicaciones y crear infraestructuras virtuales aisladas en un servidor.

Docker es un proyecto de código abierto con el que se creó contenedores que se denominan máquinas virtuales ligeras que serán menos exigentes en cuanto a recursos de hardware y software, una de las características de estos contenedores es brindar portabilidad, ligereza y autosuficiencia a las aplicaciones que se desplieguen utilizando contenedores Cloud.

La característica principal de docker es la de brindar la opción de crear infraestructuras virtuales, esto depende de las aplicaciones que vayamos a desplegar, conjuntamente con esta herramienta se usó a Bitnami como un complemento de docker el cual ofrece soporte a docker, Bitnami facilita crear Dockerfiles que contienen instalaciones independientes de herramientas necesarias para el alojamiento de aplicaciones, de este modo también se logró realizar el aislamiento de aplicaciones sin dependencias del sistema anfitrión.

También se realizó una aplicación de prueba en java server faces (JSF) que conjuntamente con la base de datos PostgresSQL demostró que la herramienta docker funciona y además brinda un despliegue de aplicaciones rápida, optima y segura a comparación de una instalación virtual tradicional.

Al finalizar dicho estudio se realizó una propuesta a la plataforma Cloud FICA con el objetivo de implementar esta tecnología y aprovechar los beneficios que brinda la herramienta docker en un servidor real.

Abstract

This project was conducted using cloud-based concepts and programming,with wich a study was conducted the cloud containers,with the finality the manage applications and create virtual infraestructures isolated on a server.

Docker is an open source project with which it was created containers that are called light virtual machine which they are less demanding as to resources the harware and software,one of the characteristics of these containers is to provide portability,lightness and self-sufficiency to applications that are deployed using Cloud containers.

The principal characteristic the Docker it is to provide the option the create virtual infraestructures, this depends on the applications that wiil be deployed ,together with this tool was used Bitnami as a complement docker which offers support to docker,Bitnami facilitates create Dockerfiles that contain independent instalations the tools needed to deploy aplications,also it was achieved make the application isolation dont have dependency the anfition system.

It was also performed a test application on Java Server Faces(JSF) that together with the data base PostgresSQl he showed that this tool it works and also provides a deploy the aplications rapid, optimal and safe unlike the tradicional instalation.

At the end of the study It has been made a proposal to the plataform Cloud Fica with the objetive of implementing this technology and take advantage the benefits that providing Docker tool in a real server.

Glosario

Introducción

En esta sección se definen las terminologías usadas en este documento, con la finalidad de dar un significado a las expresiones técnicas mencionadas en varias secciones del mismo con la intención de que el lector comprenda lo que se quiere comunicar.

Organización del glosario

Está organizado de forma ascendente según los términos y el orden alfabético tradicional en español.

Abreviaturas y definiciones

Abreviaturas

AWS. - Amazon Web Services**.**

IaaS. -Infraestructura como servicio.

ERP. - Enterprise Resource Planning.

PaaS. - plataforma como servicio.

RF. - Requisitos funcionales**.**

RFN. -Requisitos no funcionales.

SaaS. -Software como servicio.

S.O. - Sistema operativo.

VPS. - Servidores virtuales privados.

IEEE. - The Institute of Electrical and Electronics Engineers

Java EE.- Java Enterprise Edition.

JSF. – Java Server Faces.

TI.- Tecnologías de la información.

Definiciones

Apache-JMeter. - Es una herramienta de software de código abierto que permite medir la conducta funcional y el rendimiento de páginas web.

Bitnami. – Es una biblioteca de instaladores de paquetes de software para aplicaciones web.

Bootsfaces. -Es una librería que integra una fusión entre las librerías Primefaces y Bootstrap con la intención de hacer más atractivas las páginas web.

Docker. – Es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores cloud.

DockerToolbox. - Se refiere a un paquete de instalación que proporciona docker para realizar la instalación en Windows.

Dockerfiles. - Son archivos de texto plano que contienen instrucciones de instalación para crear imágenes de Docker.

Framework. - Es una estructura tecnológica de soporte definido con módulos concretos de software que pueden servir de base la organización y desarrollo de software.

GitHub. – Es una plataforma de desarrollo para software colaborativo.

Open Source. - Se refiere a una organización que dedicada a la promoción de código abierto es decir software libre.

Primefaces. -Es una librería de componentes que se adaptan a JSF.

PostgresSQL. – Se refiere a una base de datos relacional de código abierto.

SCRUM. – Metodología ágil de desarrollo de software.

Tomcat. –Es un servidor web de código abierto que permite alojar aplicaciones web

CAPITULO 1

Introducción

2.1 PROBLEMA

2.1.1 ANTECEDENTES.
2.1.2 SITUACIÓN ACTUAL.
2.1.3 PROSPECTIVA.
2.1.4 PLANTEAMIENTO DEL PROBLEMA.

2.2 OBJETIVOS

2.2.1 OBJETIVO GENERAL.
2.2.2 OBJETIVOS ESPECÍFICOS.

2.3 ALCANCE

2.4 JUSTIFICACIÓN

2.4.1 JUSTIFICACIÓN TEÓRICA.
2.4.2 JUSTIFICACIÓN PRÁCTICA.

2.5 IMPACTOS

1.1 Problema

1.1.1 Antecedentes.

El concepto de la computación en la nube, o Cloud Computing, empezó con proveedores de servicios de Internet a gran escala como Google, Amazon AWS y otros que construyeron su propia infraestructura. De entre todos ellos emergió una arquitectura: un sistema de recursos distribuidos horizontalmente introducidos como servicios virtuales de TI escalados masivamente y manejados como recursos configurados y mancomunados de manera continua.

Hoy en día además es posible utilizar el software alojado completamente en la nube, mejorando así todos los procesos y reduciendo drásticamente los costes, al reducir las necesidades de memoria para la ejecución de procesos y el mantenimiento, incrementar la eficiencia de cualquier organización al facilitar el acceso en cualquier momento, desde cualquier punto y desde cualquier dispositivo.

1.1.2 Situación actual.

Actualmente, es posible acceder a Internet a través de una serie de dispositivos, básicamente desde cualquier lugar del mundo y de forma más segura y rápida que en el pasado. La tecnología evoluciona constantemente y con ella las mejoras en el mundo de la comunicación. Cloud Computing, una de dichas tecnologías de comunicación, facilita almacenar, gestionar, compartir y poner a disposición de datos, software, aplicaciones y servicios a través de Internet.

Algunas empresas del Ecuador entienden el termino Cloud Computing y desconocen sobre las herramientas que permiten facilitar el uso de la misma es por eso que se realiza este estudio acerca de las herramientas Cloud como contenedores para alojar aplicaciones y servicios web en la nube.

1.1.3 Prospectiva.

La computación en la nube es el futuro para alojar sus aplicaciones y servicios web la misma que se ha convertido en un término renombrado en las empresas.

En plena era de Cloud Computing lo que se busca es el desarrollo de las llamadas Cloud Native Applications, que exploten todo el potencial que ofrece la nube y limite los esfuerzos a la hora de integrar tecnologías dispares.

La clave son los contenedores, entendidos como sistemas empaquetados, orientados a micro servicios y gestionados dinámicamente. La gran ventaja de los Contenedores es que permite que una aplicación y sus dependencias sean empaquetadas y operadas con menos recursos que las instancias de máquinas virtuales.

Los expertos señalan que ya hay un buen número de compañías constructoras de centros de datos investigando en la puesta en marcha de data centers basados en tecnología de contenedores. Es por eso que muchas organizaciones adoptaran estas tecnologías para mejorar la calidad de sus aplicaciones y servicios web.

1.1.4 Planteamiento del problema.

Las organizaciones tienden a usar Cloud Computing por cuestiones tecnológicas y principalmente económicas, pero desconocen las herramientas que facilitan el uso de la misma.

Muchas organizaciones tienen definida una plataforma Cloud para alojar aplicaciones y servicios web, pero ¿Cómo crear una infraestructura virtualizada aislada dentro de una plataforma web? En este punto se define el término de contenedores Cloud, estas son herramientas que permiten crear infraestructuras web para alojar aplicaciones y servicios e incluso facilitar la migración de aplicaciones entre nubes.

En la Facultad FICA de la Universidad Técnica del Norte gestiona actualmente una plataforma Cloud en la que se desconoce el uso de contenedores que permitan crear una infraestructura virtualizada aislada, este estudio permitirá implementar contenedores en la plataforma Cloud FICA de la UTN.

1.2 Objetivos

1.2.1 Objetivo general.

Elaborar un estudio sobre las herramientas Docker y Bitnami, para comprobar su funcionalidad y despliegue de aplicaciones web.

1.2.2 Objetivos específicos.

- Estudiar las características de Docker su funcionalidad y sus limitaciones como software libre.
- Crear una infraestructura virtualizada para desarrollar aplicaciones web.
- Demostrar la automatización y despliegue de aplicaciones al usar una infraestructura virtualizada.
- Elaborar una aplicación que realice una prueba de los contenedores Cloud y posiblemente utilizar imágenes Bitnami para el desarrollo de la misma.
- Comprobar si los contenedores Cloud ofrecen agilidad en las aplicaciones y servicios web.

1.3 Alcance

Este estudio se llevará a cabo con las herramientas Docker y Bitnami (versión Gratuita) para determinar su funcionalidad y características donde será necesario realizar la virtualización, el desarrollo, prueba de las herramientas y análisis de rendimiento que sea necesario con el desarrollo de una aplicación.

Contenedor Docker. – Con esta herramienta se realizará la instalación de DockerToolbox la cual se deberá instalar localmente en el ordenador esta instalación tendrá el contenedor que básicamente es la instalación del sistema operativo Linux que estará inicializada en virtual box.

Figura 1. Iconos Docker en Windows

El siguiente paso será crear un Dockerfile dentro de una carpeta que contenga todo el proyecto en el cual se construirá imágenes Docker.

Un Dockerfile describe los elementos de software que componen una imagen. Sin embargo, un Dockerfile puede describir qué entorno usar o qué comandos puede ejecutar en el contenedor. De la misma forma desplegaremos imágenes de Dockerfile las cuales podrán contener bases de datos, un servidor web etc.

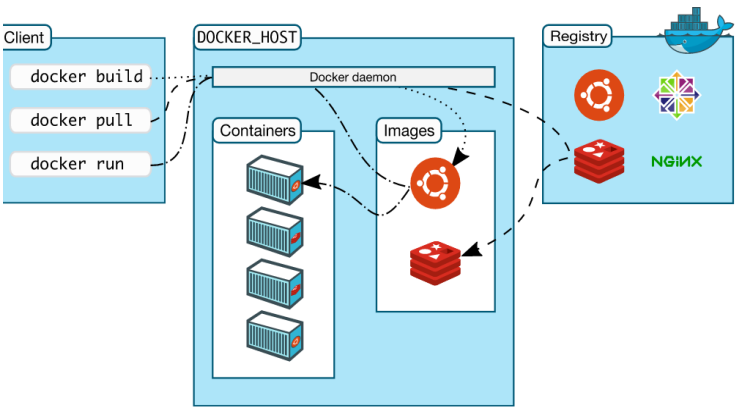


Figura 2. Arquitectura de las herramientas a usar

Bitnami. - Esta herramienta ofrece una biblioteca de aplicaciones de servidor populares y entornos de desarrollo que se puede instalar ya sea en su computadora portátil, en una máquina virtual o alojado en la nube.

Con Bitnami se usará como herramienta para utilizar con Docker y se desarrollará una aplicación web.

Figura 3. Imágenes Docker-Bitnami 1

Finalmente propondré la implementación de un contenedor en la plataforma Cloud FICA para demostrar su funcionalidad.

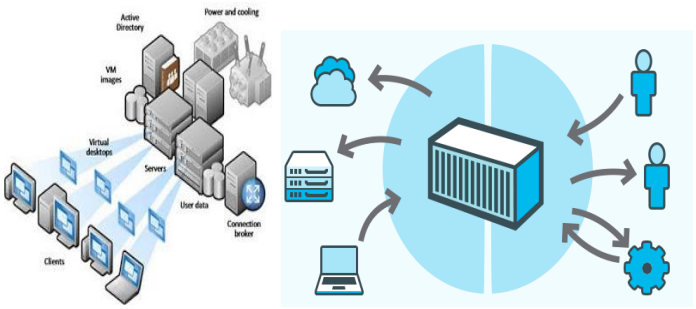


Figura 4. Arquitectura Docker

1.4 Justificación

A medida que va aumentando la tecnología se han implementado una serie de herramientas y servicios nuevos y entre ellos está Cloud Computing, tecnología que ha sido adoptada por muchas empresas al momento de alojar aplicaciones y servicios web.

Muchos desarrolladores tienen conocimiento de Cloud Computing y herramientas que facilitan su uso, pero desconocen que herramienta utilizar, ¿Qué beneficios nos brindan?, ¿cómo sería beneficioso para las entidades que la usen?

Muchas organizaciones han implementado plataformas Cloud, pero desconocen formas ágiles de desplegar aplicaciones web en este estudio propondré hacer uso de la herramienta Docker con la finalidad de virtualizar infraestructuras para el despliegue de aplicaciones y servicios web. Este estudio de contenedores Cloud permitirá sugerir a los desarrolladores de aplicaciones optar por el uso de estas tecnologías.

1.4.1 Justificación teórica.

Estas Herramientas Cloud se han consolidado en los diferentes sectores económicos, educativos y sociales, se los denomina así porque brindan a las organizaciones eficiencia económica y estabilidad en el desarrollo de aplicaciones y servicios web.

Al contar con este estudio de las herramientas Docker y Bitnami pretendo incentivar a las organizaciones a usar dichas herramientas para desplegar aplicaciones. De esta forma las organizaciones tendrán un ahorro económico significativo y una eficiencia en el desarrollo de aplicaciones y servicios web.

1.4.2 Justificación práctica.

En la actualidad, los dispositivos y herramientas acceden a servicios de software simultáneamente, creando así exceso de concurrencia de usuarios lo que hace que las aplicaciones pierdan solidez y a su vez causa molestias a los usuarios.

Con este estudio se evaluarán las funcionalidades y uso de contenedores Cloud con el objetivo de conocer sus características en el despliegue de aplicaciones web y eliminar la caída de las aplicaciones.

1.5 Impactos

Este estudio permitirá a los administradores y desarrolladores de aplicaciones realizar una infraestructura virtual que permita aislar aplicaciones de tal forma que los despliegues de software se logren hacer en cualquier lenguaje de programación que se encuentre dentro de las instalaciones independientes de Bitnami.

Además de gestionar las aplicaciones que se alojen en los contenedores permitirá un rendimiento único al ejecutar las aplicaciones debido a que la estructura de docker es súper ligera y compacta debido a que usa lo estrictamente necesario en recursos y librerías.

Permitirá a los administradores de una data center implementar esta tecnología sin mayores problemas y sin costes ya que la herramienta es libre (Open Source).

Esta tecnología eventualmente podrá suplantar a un centro de datos normal en producción ya que muchas entidades del mundo lo han adoptado y verifican su excelencia en cuanto a rendimiento y fluidez en sus aplicaciones.

CAPITULO 2

Marco teórico

2.6 CLOUD COMPUTING

2.7 ESTUDIO DE LAS HERRAMIENTAS CLOUD

2.8 VENTAJAS Y DESVENTAJAS

2.9 VIRTUALIZACIÓN

2.10 SERVIDORES VIRTUALES

2.11 CENTROS DE DATOS VIRTUALES

2.12 CONTENEDORES EN LA NUBE

2.1 Cloud Computing

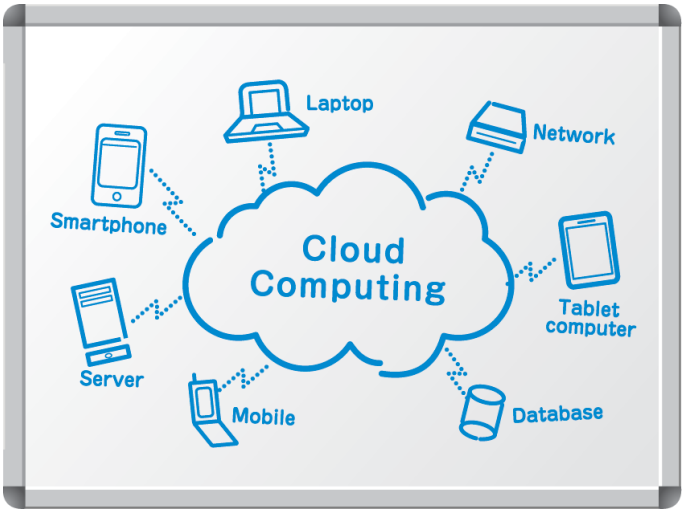


Figura 5. Cloud Computing.

Fuente: *(Livingston, 2014)*

Según (Aguilar, 2012): La nube o computación en la nube (Cloud Computing) es la plataforma tecnológica por excelencia de la década actual y, posiblemente, del futuro de la computación. Cloud Computing se ha convertido en el término de moda de todos los medios de comunicación a nivel mundial. Los desarrolladores, organizaciones y empresas analizan el nuevo modelo, sus tecnologías, sus herramientas y los proveedores, junto a toda la infinidad de aplicaciones en los numerosos campos donde ahora tienen un gran impacto: tecnológicos, económicos y sociales. (p.131)

Según la IEEE Computer Society, computación en la nube es un paradigma en el que la información se almacena de manera permanente en servidores en Internet y se envía a memorias temporales del cliente, lo que incluye computadores portátiles, equipos de escritorio, centros de

ocio, teléfonos celulares, etc. Como consecuencia se ha creado un nuevo modelo de prestación de servicios de negocio y tecnología, basados en la web. Este modelo permite al usuario acceder a un catálogo de servicios estandarizados y responder a las necesidades de su negocio. El usuario, a cambio, paga únicamente por el consumo efectuado. Los tipos de servicios que se pueden proveer a través de la nube son muy variados: almacenamiento de documentos y datos, los usuarios pueden obtener un CPU sin comprar equipo, utilizar un software para la planificación de los recursos empresariales (ERP) sin necesidad de comprarlo.(Profesional, Diego, Arévalo, Francisco, & Martín, 2011)

En resumen, la computación en la nube consiste en la posibilidad de ofrecer servicios a través de internet considerándola así una tecnología nueva que han adoptado muchas empresas a lo largo del tiempo de su aparición. Esta tecnología busca tener todos nuestros archivos en internet sin depender de la capacidad de almacenamiento que se necesite. Además, Cloud Computing permite alquilar una infraestructura de hardware en la red con herramientas IaaS (Infraestructura como servicio) también podemos hacer uso de herramientas colaborativas y de desarrollo conocidas como PaaS (plataforma como servicio) y si deseamos consumir aplicaciones ofrecidas por el proveedor de servicios o pertenecientes a la propia empresa de tal manera que permita ofrecer servicios online avanzados a sus clientes entonces utilizaremos las herramientas SaaS (software como servicio).

La computación en la nube ciertamente engloba varios aspectos tecnológicos como la virtualización, la computación distribuida, redes, servicios de software y servicios web

de esta manera se define que Cloud Computing no es un desarrollo revolucionario reciente sino el resultado de evolución de varias tecnologías. (Sosinsky, 211)

2.2 Estudio de las herramientas Cloud

Como se ha mencionado, Cloud Computing es un modelo tecnológico de prestación de servicios a través del internet, el cual ha brindado almacenamiento sin limitaciones y soporte a muchas empresas. Cloud Computing se clasifica en tres niveles que son las herramientas IaaS (Infraestructura), PaaS (Plataformas), Apps y servicios (Aplicaciones y servicios).

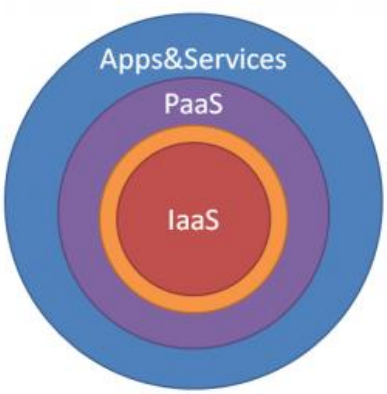


Figura 6. Tres niveles de Cloud Computing.

- **IaaS** (infraestructura como servicio) este permite determinar el almacenamiento y capacidad de computo con las técnicas de virtualización.
- **PaaS** (Plataformas como servicio) es un sistema que puede ser programado y personalizado por los desarrolladores," si se puede programar se trata de una plataforma caso contrario, no los es" existen varias plataformas Cloud como Amazon, Google Cloud Plataform, Windows Azure, Openshift, Cloud Foundry, Openstack entre otros.
- **AppS&Services** (aplicaciones y servicios) se trata de las aplicaciones como servicio que son desplegadas en los niveles IaaS y PaaS.

Este estudio contiene el nivel IaaS, Apps y servicios ya que se realiza una infraestructura virtualizada además de desplegar aplicaciones en la misma y adicionalmente Docker podrá ser

instalado en una plataforma web es decir cumplirá con los 3 niveles de Cloud Computing, esta nueva propuesta tecnología permite a los desarrolladores aislar infraestructuras de software en un lenguaje propio independientemente del que ya se encuentre instalado en el sistema anfitrión.

2.3 Ventajas y desventajas

La computación en la nube es una forma de manejar la información que está tomando fuerza en estos últimos años, es por eso que es muy importante conocer las ventajas y desventajas que trae. Una de las ventajas más importantes es el acceso a la información y los servicios desde cualquier lugar es decir la disponibilidad. Y por consecuente una desventaja es el acceso de toda la información a terceras empresas a no ser que se tenga una nube privada o plataformas Cloud propias.(José, Condori, & Stallman, 2011)

Tabla 1 *Ventajas y desventajas de usar Herramientas Cloud*.

CLOUD COMPUTING	
Ventajas	**Desventajas**
Accesibilidad.- ofrece el acceso de información en cualquier parte del mundo y a cualquier hora.	**Dependencia.-** siempre el usuario dependerá del proveedor del servicio de Cloud Computing a no ser que se tenga una plataforma Cloud propia.
Bajos costos.- no se necesita equipos de alta potencia o de alto precio para procesar aplicaciones basadas en web que están en la nube .dado que las aplicaciones se ejecutan en la nube y no en un ordenador de escritorio.	**Conexión a internet.-** la mayor desventaja se podría decir que es la conexión a internet debido a que para acceder a los datos alojados en la nube se requiere la conexión a internet con un buen ancho de banda.

Espacio de almacenamiento. - básicamente Cloud Computing ofrece una capacidad de almacenamiento a gran escala dependiendo del proveedor y el contrato del servicio.	**Migración.-** No es fácil transferir grandes volúmenes de datos entre nubes.
Facilidad de gestión de datos de información. - dado que los datos están en una sola ubicación se podrá manejarla y organizarla fácilmente.	Todo lo que está en la red puede ser susceptible de ser utilizado por alguien no autorizado accediendo a datos o información posiblemente de gran valor. (Peña, 2013)
Diversidad de dispositivos.- se podrá acceder a nuestros datos no solo desde un ordenador sino desde cualquier otro dispositivo con internet como iPad, portátil, tabletas, Smartphone.	Cifrar las conexiones es decir las conexiones que se realicen con el ISP deben ser con el protocolo HTTPS para garantizar la seguridad. (Peña, 2013)

2.4 Virtualización

La solución tradicional para una data center es instalar un sistema operativo estándar, individual y convencional que asegure los recursos compartidos y proteja las aplicaciones brindándole seguridad y optimización a las mismas.

La alternativa a esta solución es usar recursos virtualizados que realicen funciones de un data center. La virtualización es el principio de Cloud Computing con lo cual se puede simplificar recursos y tareas administrativas. (Marinescu, 2013)

Virtualización es el uso de los recursos de las computadoras para simular o imitar a otros recursos de ellas o las propias computadoras en su totalidad; también es un mecanismo fundamental para la entrega de servicios; proporciona una plataforma para optimización

de recursos de TI complejos en un modo escalable (crecimiento de recursos de modo eficiente y adaptable a los servicios de entrega).

Sin virtualización es muy difícil de gestionar o administrar recursos lógicos y físicos es por eso que se considera importante la virtualización porque permite simplificar muchos aspectos de la computación, además de facilitar la creación de servidores virtuales, almacenamiento virtual, redes virtuales entre otros.

La virtualización es una de las tendencias claves dentro de los centros de datos privados o públicos y está considerada como una de las tecnologías que cambiará los entornos de TI. Un administrador de TI podrá dividir (particionar) un servidor físico en diferentes servidores virtuales, en el que no se distinguirá si un servidor es físico o real.

El concepto de virtualización según Intel se define como la abstracción de hardware del computador que oculta el computador físico de cómo se utiliza. Con la virtualización un servidor físico aparece como múltiples “máquinas virtuales” lógicas utilizando un hipervisor o software de monitor de máquina virtual (VMM, Virtual Machine Monitor) de este modo la plataforma del servidor virtualizado permite que múltiples sistemas operativos se ejecuten sobre un sistema anfitrión (host). (Aguilar, 2012)

La virtualización se puede aplicar en varios elementos computacionales como:

- Memoria
- Red
- Almacenamiento

- Hardware
- Sistemas operativos
- Aplicaciones

Estos elementos son importantes y hacen posible el uso de Cloud Computing (Hurwitz, Bloor, & Kaufman, 2010)

¿Cómo virtualizar?

Para empezar a virtualizar se debe contemplar, al menos los siguientes aspectos, el primer paso es realizar una evaluación del entorno en el que se desea virtualizar se debe considerar el CPU, memoria, capacidad del sistema de archivos, espacio de disco total, adaptadores y otros dispositivos.

El segundo paso se refiere a el software de virtualización que son muy numerosos, pero para este estudio se identificó a los más solicitados por los usuarios y de versión gratuita.

VMWARE.

Existen dos versiones estas son el VMware Player(gratuito). Están limitados sus recursos y el VMware Workstation que es uno de los programas más populares y reconocidos en el campo de virtualización (Software con licenciamiento).

Figura 7. Icono vmware.

Esta herramienta permite crear máquinas virtuales para una gran variedad de sistemas operativos (tales como Windows, Mac, Linux, Solaris entre otros) tiene una capacidad de

virtualizar hasta 2TB de discos virtuales, asignar hasta 8 procesadores virtuales por máquina, 64 GB de memoria RAM por máquina, también tiene la posibilidad de conectarnos en forma remota desde otro computador, teléfono inteligente o Tableta sin tener acceso a la maquina anfitrión.

ORACLE.

Hasta el momento existen 4 versiones como lo es VirtualBox (Gratuito y software de código abierto "Open Source"), Citrix (Xen). uno de los programas de virtualización más populares, Paralles (con licenciamiento), Red Hat.

Figura 8. Icono Virtual box.

Para este estudio se realizó con la herramienta VirtualBox el cual es un programa gratuito y de código abierto creado por Sun/Oracle; el cual puede usarse en sistemas anfitriones multiplataforma. (Aguilar, 2012)

2.5 Servidores virtuales

Figura 9. Servidores Virtuales.

La virtualización de servidores se encuentra en la actualidad en una de las facetas más importantes dentro de las tendencias de modernización e implementación de nuevas tecnologías. En estos sistemas incluyen la virtualización del almacenaje, red y control de carga de trabajo. Los servidores virtuales consisten en agrupar diferentes aplicaciones y servicios de sistemas heterogéneos dentro de un mismo hardware, de modo que los usuarios y el mismo sistema lo vean como maquinas independientes dedicadas.(Servidores et al., n.d.)

En este estudio se pretende virtualizar servidores privados (VPS) que brindan una solución perfecta cuando se trata de crear un servidor propio es decir se virtualizará un sistema operativo en este caso será Ubuntu en la versión 14.04 que contendrá el contenedor Cloud Docker demostrando su funcionalidad.

El uso de los VPS en recomendado ya que donde se aplique esta tecnología se necesitará garantizar los recursos tanto de hardware como de software y lo más importante la seguridad de los datos que se almacenen en las aplicaciones.

2.6 Centros de datos virtuales

Figura 10. Docker aplicado a centros de datos virtuales.

Hoy en día todas las empresas necesitan infraestructuras para alojar sus aplicaciones, base de datos entre otros; con el fin de almacenar datos y organizarlos, muchas empresas crean o alquilan

infraestructuras, las entidades que deciden crear infraestructuras deben elegir que equipamiento comprar según las necesidades de la empresa, y la que alquila la infraestructura deberá conocer al proveedor que brinda el servicio en base a las certificaciones y beneficios que brinda la misma.

En este estudio se realizó la virtualización del contenedor Docker el cual brinda la opción de crear infraestructuras virtualizadas aisladas con la finalidad de poder desplegar aplicaciones según lo necesitemos, es decir tendremos la posibilidad de crear centros de datos virtuales privados.

Ventajas de un centro de datos virtual.

- Menores recursos físicos
- Reducción de costes
- Potencia de almacenamiento y procesamiento
- Fluidez en aplicaciones desplegadas
- Menos acumulación de calor en los equipos
- Backups más fáciles
- Recuperaciones inmediatas
- Menos contaminación

2.7 Contenedores en la nube

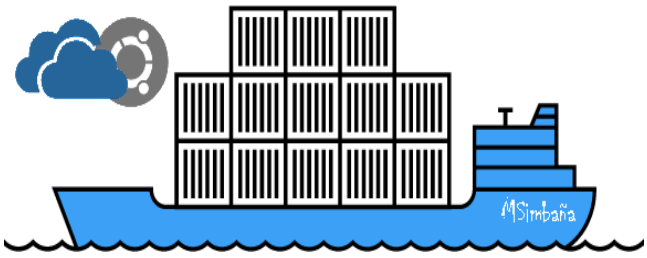


Figura 11. Contenedores Docker.

Esta nueva tendencia tecnológica de virtualizar contenedores brinda soluciones optimas al administrador de sistemas y adicionalmente al desarrollador de aplicaciones al momento de crear infraestructuras virtualizadas.

Un contenedor es simplemente es un proceso para el sistema operativo que, internamente, contiene la aplicación que queremos ejecutar y todas sus dependencias usando indirectamente el kernel del sistema operativo que se esté usando.

Al realizar un paralelismo entre el contenedor y una máquina virtual se define que ambos son sistemas autocontenidos que tienen una gran diferencia en que una máquina virtual necesita contener todo el sistema operativo mientras un contenedor aprovecha el sistema operativo en el que se ejecute.

Ejemplo:

Se desea virtualizar una base de datos en una máquina virtual y en un contenedor Docker, establecer los requisitos necesarios para la virtualización de la BDD.

Tabla 2 *Diferencia Máquina Virtual y un Contenedor Docker*.

Máquina Virtual	**Contenedor Docker**
• Una máquina física que aporte el hardware • Un sistema operativo "Host" sobre esta máquina física • Un sistema de virtualización o hypervisor que gestione las peticiones al hardware virtual y las ejecute sobre el real • Un sistema operativo "Guest" bajo el hypervisor. Este sistema debe ser completo ya que no puede obtener recursos del kernel de su Host • Instalar bajo el sistema "Guest" el motor de base de datos y todas sus dependencias	• Una máquina física o virtual • Un sistema operativo sobre esta máquina • El motor de Docker instalado en esta máquina • Un contenedor basado en una imagen que contenga el motor de base de datos y todas sus dependencias

Figura 12. Contenedor Docker mas BDD.

Las diferencias de hardware y software son notables como se muestra en la tabla 2, además de brindar agilidad, escalabilidad y eficiencia al usar contenedores Docker, realizaremos varios contenedores que dentro de sí tengan su propia instalación.

Al realizar este estudio de contenedores no solo podremos hacer instalaciones de base de datos sino además podremos realizar varias instalaciones necesarias para las aplicaciones que se desee desplegar según lo necesitemos.

Figura 13. Contenedores Cloud.

Se debe mencionar que además de Docker container existe otra herramienta que cumple una función similar como google container y Amazon container con los cuales eventualmente se podrá migrar container a las diferentes herramientas mencionadas. Estas herramientas son recientes por lo que la información acerca de cada una de ellas es escasa.

CAPITULO 3

Herramientas y métodos

3.1 HERRAMIENTA DOCKER

3.1.1 ELECCIÓN DEL SISTEMA OPERATIVO LINUX.
3.1.2 INSTALACIÓN DE LA HERRAMIENTA DOCKER.
3.1.3 ADQUIRIR IMÁGENES DOCKER DESDE UN REPOSITORIO GITHUB.
3.1.4 CONSTRUIR IMÁGENES DOCKER A PARTIR DE UN DOCKERFILE.
3.1.5 CREAR CONTENEDORES DOCKER.
3.1.6 ENLAZAR CONTENEDORES.
3.1.7 DOCKER HUB.
3.1.8 ACTUALIZACIÓN Y DESINSTALACIÓN DE LOS COMPONENTES DOCKER.

3.2 HERRAMIENTA BITNAMI

3.2.1 ADQUIRIR IMÁGENES BITNAMI COMPATIBLES CON DOCKER.

3.1 Herramienta Docker

Docker es un proyecto de código abierto con el que se podrá crear contenedores que son definidos como máquinas virtuales ligeras en Linux. Además, permite crear contenedores Cloud ligeros y portables para alojar aplicaciones y desplegarlos en su propio lenguaje según las instrucciones generadas por el dockerfile.

El dockerfile es un archivo de texto plano que recibe como parámetros instrucciones de tipo Linux, es decir con este archivo se podrá instalar dependencias de las aplicaciones, alojar aplicaciones con las dependencias instaladas y desplegarlas.

La principal ventaja de utilizar contenedores es que no depende de las instalaciones del sistema anfitrión es decir podremos realizar las instalaciones necesarias para desplegar aplicaciones dentro de un contenedor Cloud.

Esta herramienta también permite portabilidad en las aplicaciones quiere decir que además de crear contenedores, docker brinda la facilidad de crear repositorios en la nube que eventualmente serán alojados los contenedores, también se puede descargar estos contenedores desde otro host o plataforma Cloud que tenga instalado Docker y desplegarlo sin ningún problema.

Docker carece de todo un sistema completo, sino únicamente usa aquellas librerías, archivos y configuraciones necesarias para desplegar las funcionalidades que contenga. Asimismo, Docker se encarga de gestionar el contenedor y las aplicaciones que contenga, dicho esto comprobamos que esta herramienta es autosuficiente.

Características y funcionalidades de Docker.

- Autogestión de los contenedores.
- Fiabilidad
- Aplicaciones libres de las dependencias instaladas en el sistema anfitrión.

- Capacidad para desplegar varios contenedores.
- Contenedores muy livianos que facilitan su almacenaje, transporte y despliegue.
- Capacidad para desplegar una amplia gama de aplicaciones.
- Compatibilidad Multi-Sistema es decir que podremos desplegar nuestros contenedores en múltiples plataformas.
- Podremos compartir nuestros contenedores a través del repositorio Docker Hub.

3.1.1 Elección del sistema operativo.

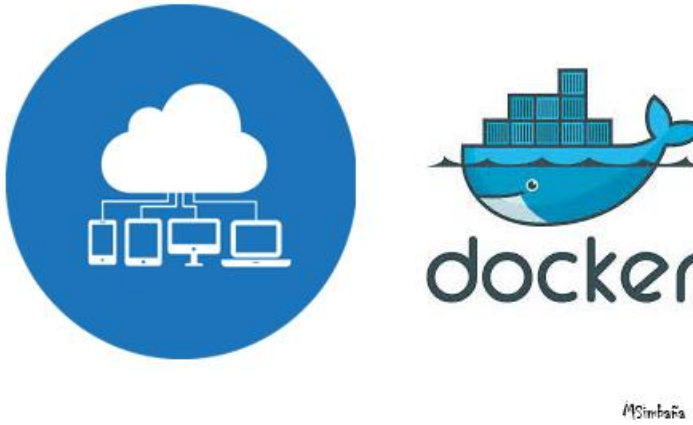


Figura 14. Docker Cloud.

Docker es una herramienta multiplataforma es decir se adapta a varios sistemas operativos ya sea en Linux, Windows o MAC además puede instalarse bajo cualquier plataforma Cloud como Openstack, Open Nebula, Eucalyptos entre otros, en este estudio se realizó la instalación bajo Linux en la versión de Ubuntu 14.04 a nivel local con el objetivo de comprobar la funcionalidad de la herramienta Docker.

La selección de este sistema operativo se estableció debido a que Ubuntu es una versión de Linux amigable con el usuario, esto permitió adentrarse más en los conceptos de Docker e instalar la herramienta de una forma rápida y eficaz.

Requisitos mínimos para la instalación de Docker.

Tabla 3 *Requisitos mínimos de instalación.*

Windows	Linux	Mac OS
• Para instalación de Docker se necesita contar con Windows 7 x64 en adelante. • Adicionalmente dirigirse a la página oficial de Docker y descargaremos los componentes necesarios para su instalación.	• Para instalar Docker en Linux se ha tomado tres S.O usados frecuentemente como son Centos, Ubuntu, Fedora. • En Centos debe contar con la versión 7x64 en adelante. • En Ubuntu se debe contar con la versión 14.04 x64 en adelante. • En Fedora se debe contar con la versión 22 x64 en adelante.	• Para instalar Docker se debe contar con OS X 10.8 “Mountain Lion”. • Adicionalmente dirigirse a la página oficial de Docker y descargaremos componentes necesarios para su instalación.
Requisitos mínimos de hardware		
• Memoria RAM de 1GB. • Espacio en disco de 20 GB.		

3.1.2 Instalación de la herramienta Docker.

Procedemos a ingresar al terminal de Ubuntu para proceder a ejecutar las instrucciones de instalación.

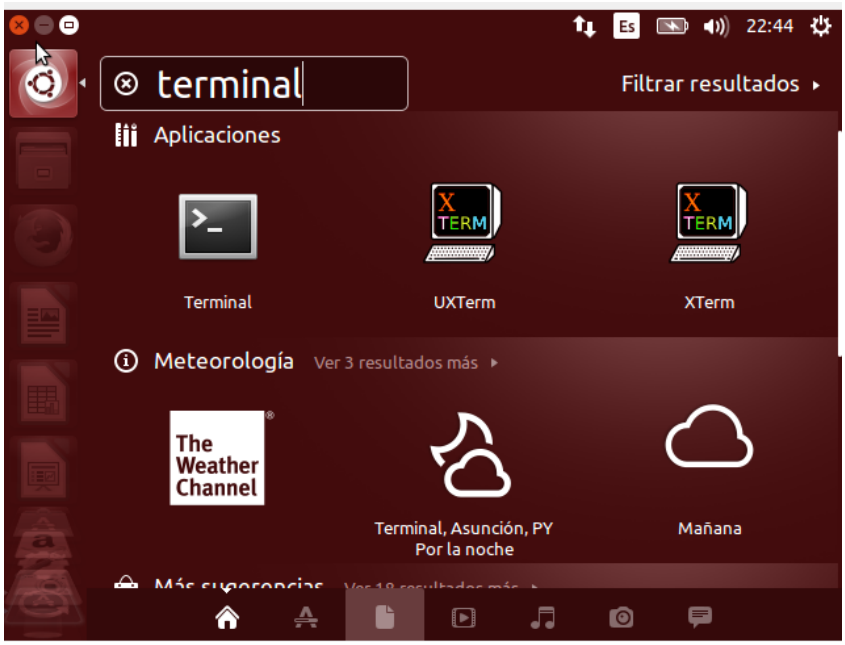


Figura 15. Terminal Ubuntu 1.

Una vez que se despliega el terminal, se ingresan las siguientes instrucciones usando el usuario (su) de instalación de Docker. **sudo apt-get -y install curl**

```
docker@docker-VirtualBox:~$ sudo su
[sudo] password for docker:
root@docker-VirtualBox:/home/docker# apt-get -y install curl
```

Figura 16. Súper usuario.

curl -sSL https://get.docker.com | sudo sh

```
root@docker-VirtualBox:/home/docker# curl -sSL https://get.docker.com | sudo sh
```

Figura 17. Instalación curl para docker.

Al ejecutar la instrucción anterior se muestra la siguiente ventana.

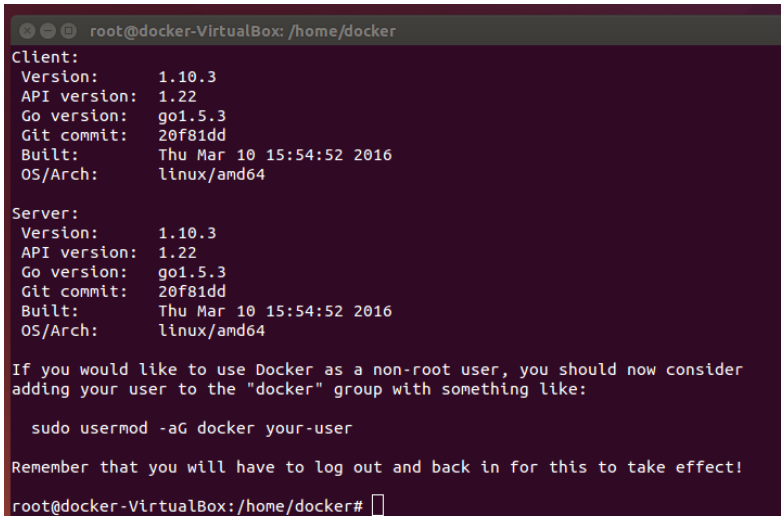


Figura 18. Verificación de curl instalado.

Luego se asignar permisos de docker al usuario Ubuntu, en este caso se asigna los permisos al usuario Ubuntu

docker. sudo usermod -a -G docker user

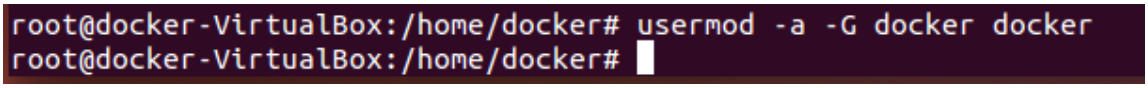


Figura 19. Conceder permisos a Docker.

Se debe verificar si se ha descargado los paquetes de instalación Docker con las siguientes instrucciones en el terminal de Ubuntu.

docker –v

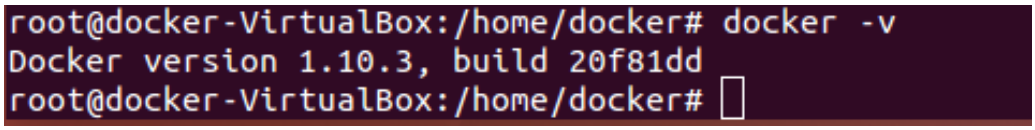


Figura 20. Versión de Docker.

Una vez instalado la herramienta procedemos a ejecutar el siguiente comando con la finalidad de mostrar las características en cuanto a versión y memoria en uso de Docker.

docker info

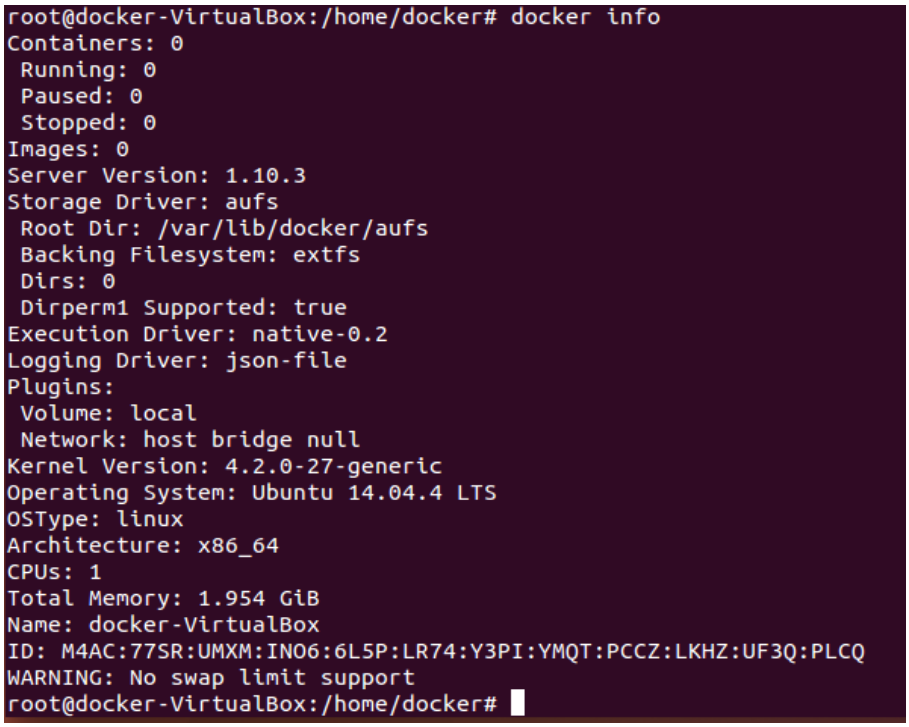


Figura 21. Información inicial de Docker.

Al completar de ejecutar las instrucciones mencionadas anteriormente finalizara la instalación de la herramienta Docker.

Tabla 4 *Comandos básicos de Docker*.

Comando básicos de Docker	
docker versión	Muestra la versión de docker instalada.
docker info	Muestra las características de docker.
docker images	Muestra imágenes de docker instaladas.

docker ps	Muestra los contenedores en ejecución.
docker pull [imagen]:[Version]	Realiza un petición de descarga de imágenes.
docker run [imagen]:[Version]	Ejecuta una imagen especifica instalada.

3.1.3 Adquirir imágenes Docker desde un repositorio GitHub.

Antes de empezar a crear y usar imágenes docker hay que tener en cuenta dos conceptos que se usa a lo largo de todo este estudio conocidas como las instrucciones de Git push y pull.con esta instrucción mencionada se puede subir y descargar información de los repositorios Git o GitHub.

Figura 22. Imagen GitHub.

En esta sección se descarga una imagen Docker de GitHub en Ubuntu, al ingresar el terminal de Ubuntu con las siguientes instrucciones.

```
docker@docker-VirtualBox:~$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
87192bdbe00f: Pull complete
28e09fddaacb: Pull complete
7e15ce58ccb2: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:f6e8757419147ba099af8cec4db365b3603472bd6182722b16fdae932c0bf3bf
Status: Downloaded newer image for ubuntu:latest
docker@docker-VirtualBox:~$
```

Figura 23. Imagen pull Ubuntu.

Luego se debe escribir el siguiente comando para ejecutar la imagen y verificar su funcionalidad.

docker run ubuntu:14.04 /bin/echo 'Hello world'

```
docker@docker-VirtualBox:~$ docker run ubuntu:14.04 /bin/echo 'Hello world'
Unable to find image 'ubuntu:14.04' locally
14.04: Pulling from library/ubuntu
87192bdbe00f: Already exists
28e09fddaacb: Already exists
7e15ce58ccb2: Already exists
a3ed95caeb02: Already exists
Digest: sha256:11211aa27bbae93283ce329637d3dd637c05f89d3e3c8a691e125b3f537dfaa1
Status: Downloaded newer image for ubuntu:14.04
Hello world
docker@docker-VirtualBox:~$
```

Figura 24. Ejecución de la imagen Ubuntu.

Para revisar que efectivamente se obtuvo la imagen Docker de Ubuntu se usa la siguiente instrucción para visualizar un listado de todas las imágenes existentes con sus respectivas características.

```
docker@docker-VirtualBox:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
ubuntu              14.04               b549a9959a66        4 days ago
188 MB
ubuntu              latest              b549a9959a66        4 days ago
188 MB
docker@docker-VirtualBox:~$
```

Figura 25. Listado de imágenes Docker instaladas.

Al concluir de ejecutar las instrucciones mencionadas anteriormente se tendrá instalado las imágenes de Ubuntu en Docker, este proceso de debe realizar para comprobar que la herramienta está instalada y lista para ser usada.

Figura 26. Ubuntu más Docker.

3.1.4 Construir imágenes Docker a partir de un dockerfile.

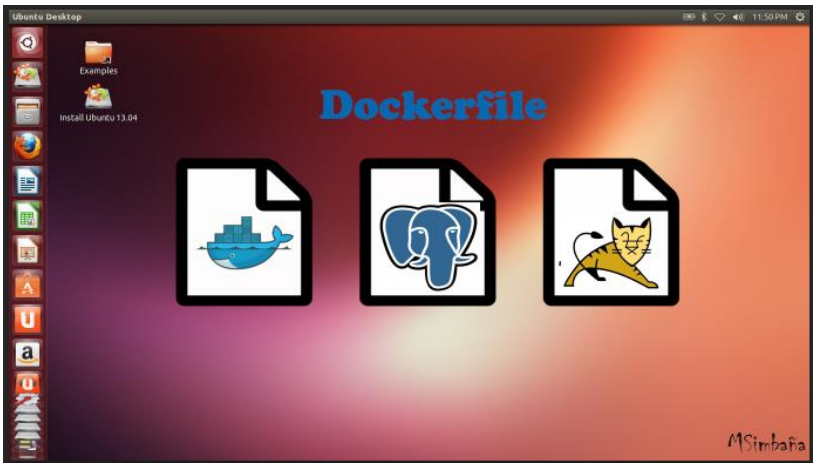


Figura 27. Imágenes Dockerfile en Ubuntu.

Para empezar con la construcción de un contenedor hay que crear imágenes Docker; en esta sección se realiza un ejemplo práctico y sencillo que es el despliegue de una página HTML en un servidor apache con el objetivo de aprender a usar esta herramienta.

Ejemplo:

Se crea una carpeta por línea de comandos con el nombre de app luego dentro de aquella carpeta se crea el dockerfile y el archivo html.

- Ingresamos al terminal de Ubuntu.

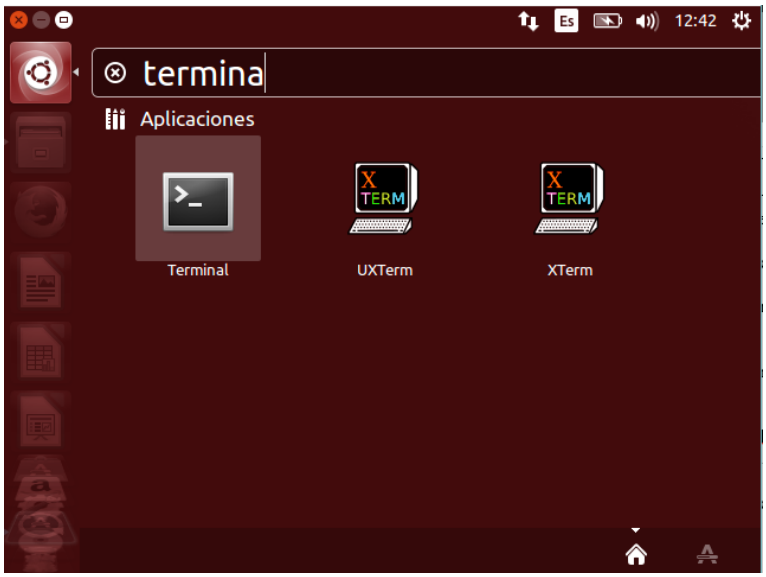


Figura 28. Ingreso al terminal de Ubuntu.

- Construimos la carpeta con el nombre app que contendrá nuestro dockerfile.

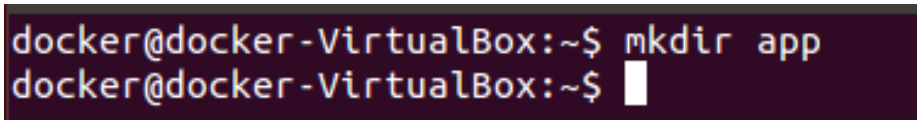


Figura 29. Crear una carpeta desde el terminal Ubuntu.

- Luego de construir la carpeta adicionalmente crearemos un archivo de texto plano que contenga las siguientes instrucciones:

```
FROM ubuntu:14.04
MAINTAINER MSimbaña

RUN apt-get update && apt-get install -y apache2

COPY hello_world.html /var/www/html/

CMD ["/usr/sbin/apache2ctl", "-e", "info", "-DFOREGROUND"]
```

Figura 30. Prueba de un dockerfile.

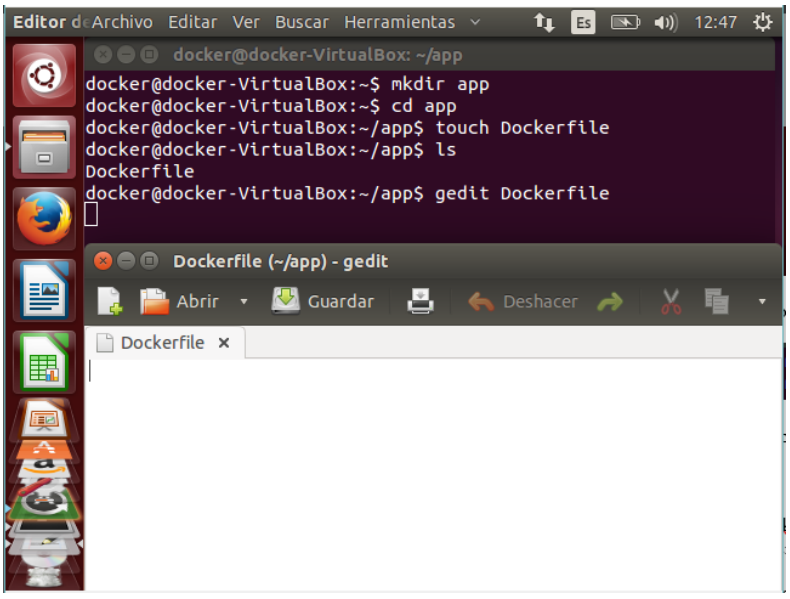


Figura 31. Construir el Dockerfile desde el terminal Ubuntu parte 1.

Un dockerfile es un archivo de texto plano que contiene instrucciones ya sea de instalación o de configuración de herramientas necesarias para desplegar aplicaciones web.

- Al finalizar con las instrucciones se tiene algo similar a la siguiente ventana:

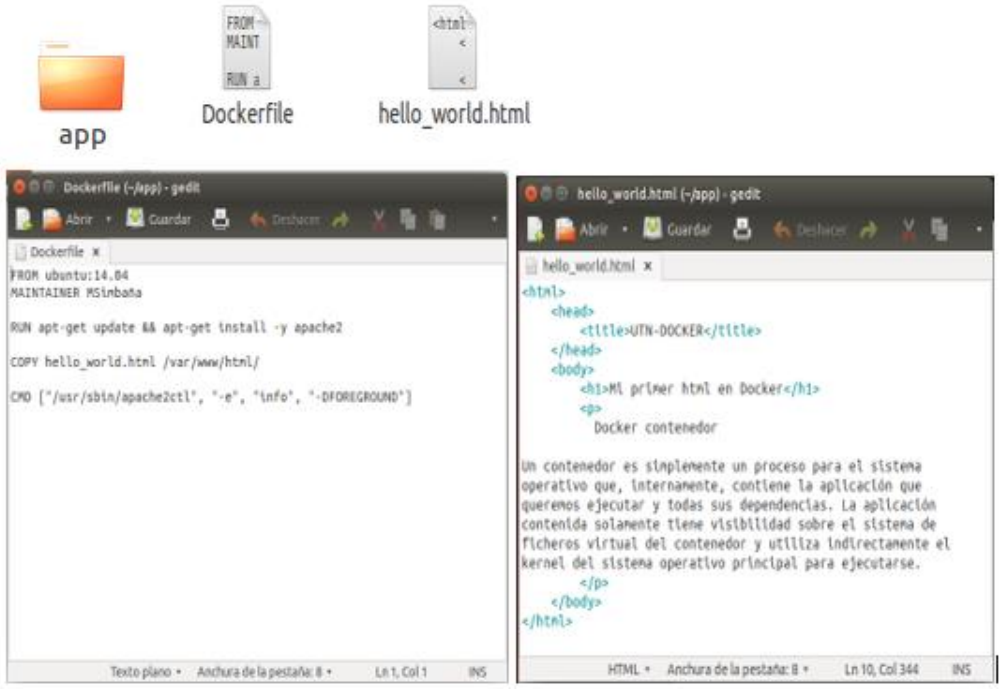


Figura 32. Construir el Dockerfile desde el terminal Ubuntu parte 2.

- Las instrucciones de este dockerfile son sencillas por lo que interpretaremos lo que se pretende realizar:

Línea 1: Usando esta instrucción le dice a docker que desde la imagen Ubuntu 14.04 que ya se descargó anteriormente realice una imagen copia.

Línea 2: En esta instrucción se especifica el autor de la imagen que se pretende crear.

Línea 3: Realiza la instalación del servidor apache dentro de la imagen docker.

Línea 4: Crea una copia de un archivo html en la ruta de instalación del servidor apache.

Línea 5: Verifica y crea directorios necesarios.

- Al tener todos estos archivos en la carpeta app se debe construir la imagen usando el dockerfile por el terminal de Ubuntu usando la siguiente instrucción.

docker build -t docker/app

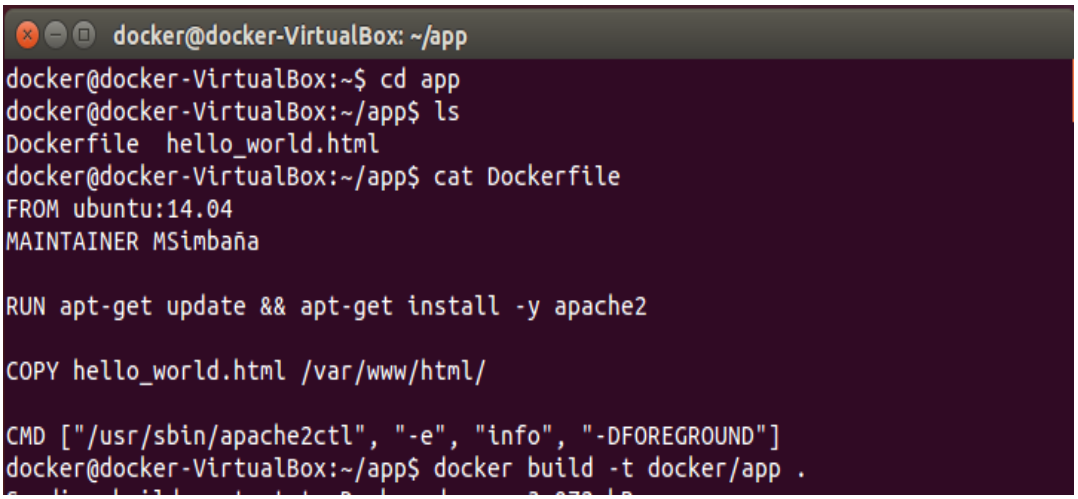


Figura 33. Construir el Dockerfile desde el terminal Ubuntu parte 3.

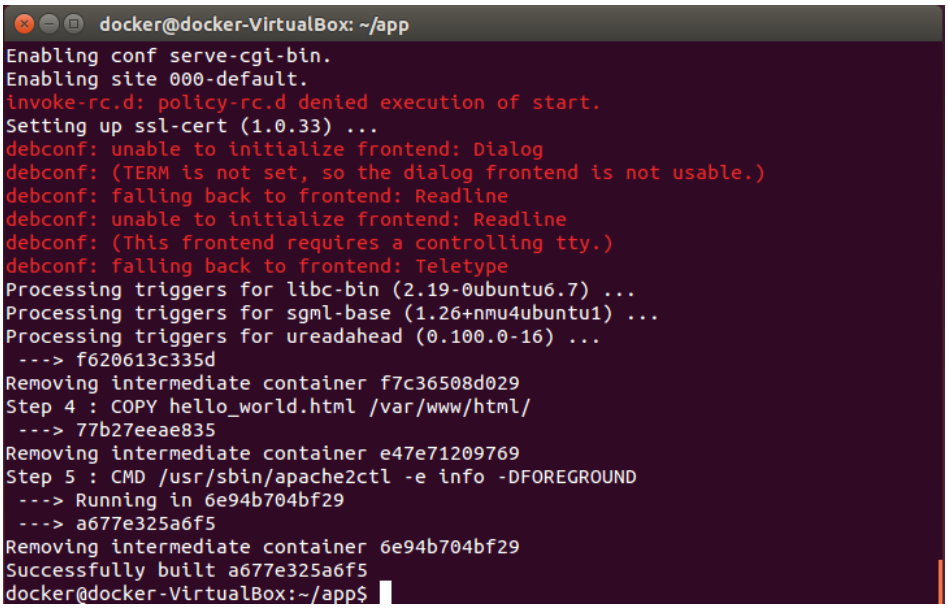


Figura 34. Finaliza la construcción del Dockerfile.

- Finalmente, para comprobar que efectivamente se construyó la imagen usando como referencia el dockerfile acudimos a la instrucción **docker images**.

```
docker@docker-VirtualBox:~/app$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
docker/app          latest              a677e325a6f5        4 minutes ago
224 MB
ubuntu              14.04               b549a9959a66        6 days ago
188 MB
ubuntu              latest              b549a9959a66        6 days ago
188 MB
docker@docker-VirtualBox:~/app$
```

Figura 35. Verificar la construcción del Dockerfile.

Hasta el momento se ha creado una imagen de autoría propia que es compatible con Docker que contiene apache y también una página HTML.

3.1.5 Crear contenedores Docker.

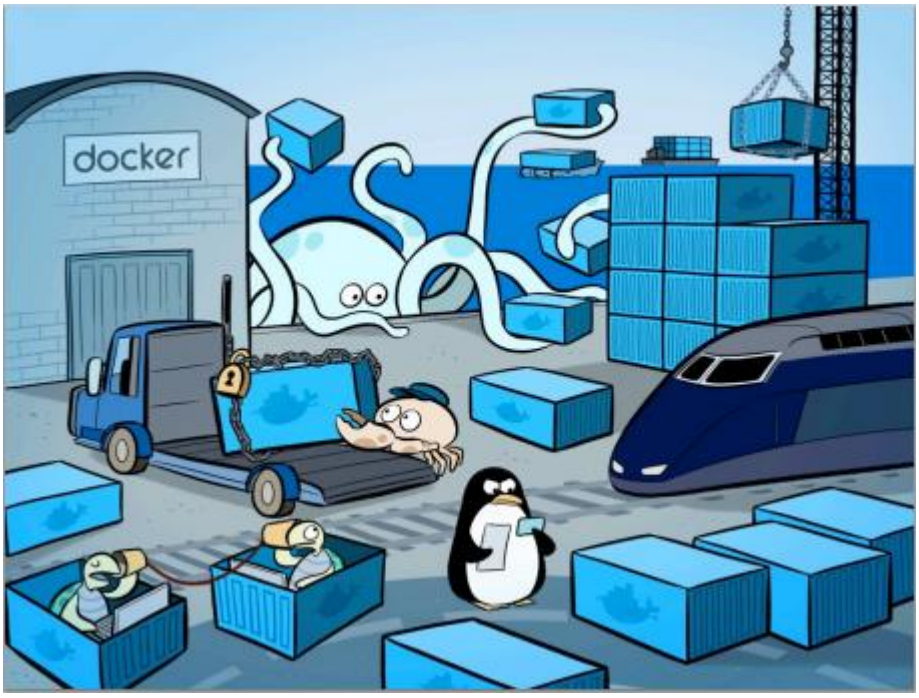


Figura 36. Logos Docker1. *(brujeador, 2016)*

El requisito principal para crear contenedores es la imagen docker como la que construimos anteriormente. Para crear el contenedor en base a una imagen Docker se debe usar la siguiente instrucción

docker run -d -p 8080:80 docker/app

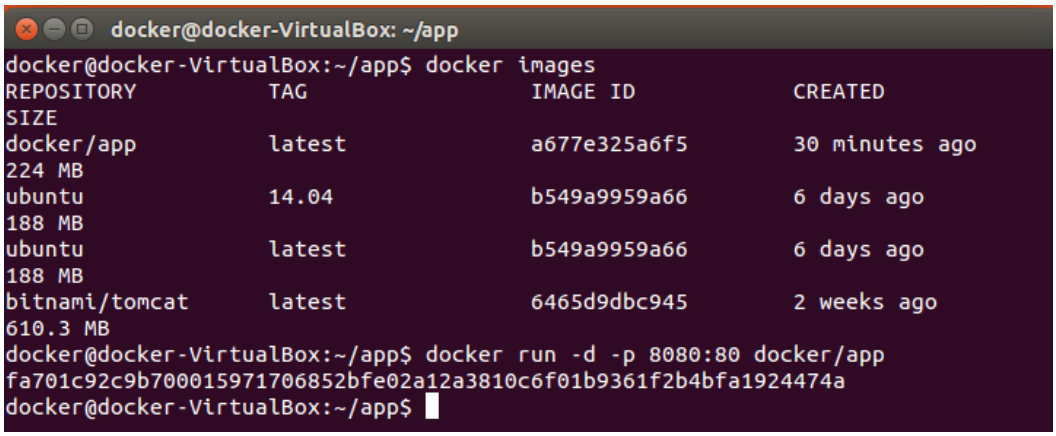


Figura 37. Ejecutar la imagen como contenedor.

Esta instrucción ejecuta la imagen para crear el contenedor, adicionalmente especifica el puerto con el que se desea hacer el despliegue junto al puerto por defecto de apache y por último el nombre de la imagen en este caso tiene el nombre de docker/app.

El siguiente paso es verificar que la imagen se convirtió en contenedor con la siguiente instrucción.

docker ps

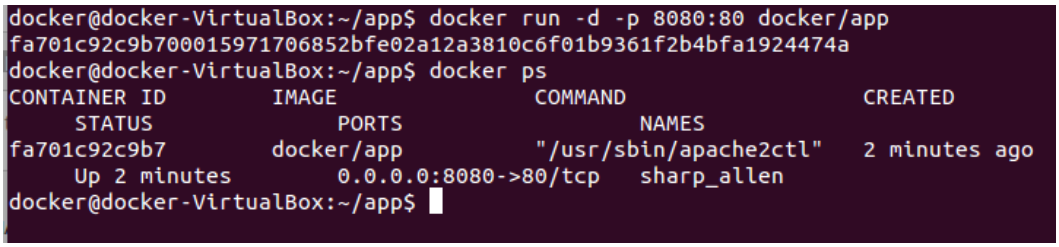


Figura 38. Lista de contenedores ejecutándose.

Luego de ejecutar las instrucciones anteriores se debe dirigir al navegador y comprobar que se realizó el despliegue; en este caso de nuestra página html usando la dirección localhost:8080/nombrehtml.

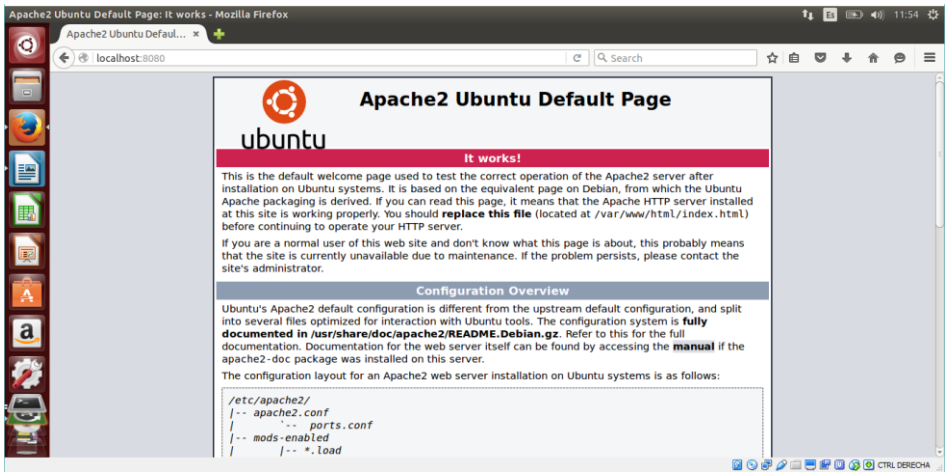


Figura 39. Contenedor con apache ejecutándose.

En esta sección usando un dockerfile de nuestra autoría hemos construido la imagen de apache que será compatible con docker y adicionalmente se ha ejecutado dicha imagen para convertirla en un contenedor que dentro de sí contendrá dicha herramienta con las configuraciones establecidas previamente por el desarrollador.

Figura 40. Despliegue de una página html en apache.

3.1.6 Enlazar contenedores.

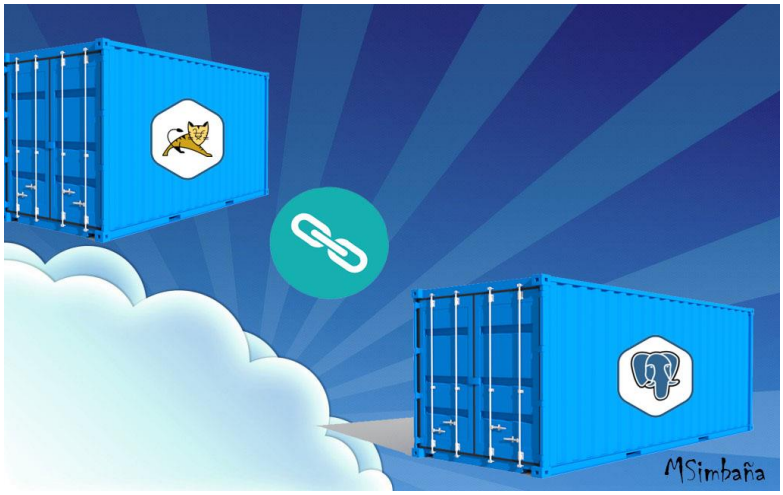


Figura 41. Enlace entre contenedores.

Después de crear imágenes docker que posteriormente se convertirán en contenedores al ejecutarlos se necesita crear enlaces para establecer la comunicación entre contenedores, en algunos casos no es necesario, pero, en el desarrollo de este estudio se usan para determinar si es funcional y comprobar si se puede establecer una conexión entre contenedores.

Al momento de crear contenedores se establecen ciertos parámetros como el puerto y el nombre con el que será conocido el contenedor a ejecutarse de tal forma que al establecer un enlace con otro lo utilice como referencia para la comunicación.

Ejemplo:

Se desea establecer una conexión entre un contenedor que contenga Tomcat y un contenedor que contenga postgres con la finalidad de que el contenedor Tomcat pueda acceder a la información del contenedor de postgres.

Previamente obtenidas las imágenes correspondientes se ejecuta las siguientes instrucciones en el servidor que contenga docker, se inicia desde la capa de la base de datos usando las siguientes instrucciones:

```
docker run -d -p 9090:5432 --name db1 -e POSTGRESQL_USERNAME=masimbaniaa -e
POSTGRESQL_PASSWORD=1003862792 -e POSTGRESQL_DATABASE=juegos_db
bitnami/postgresql:latest
```

Figura 42. Instrucción de enlace 1.

Después de haber ejecutado la instrucción de postgres podemos observar que se crea un contenedor usando el comando **docker ps.** Al ejecutar esta instrucción se establece un nombre al contenedor que es usado para crear el enlace con el contenedor Tomcat.

Finalmente se ejecutan las instrucciones siguientes para que el contenedor de Tomcat pueda establecer la conexión y además pueda extraer información del contenedor de postgres.

```
docker run -d -p 8090:8080 --name app1 -e TOMCAT_USERNAME=admin
-e TOMCAT_PASSWORD=admin --link db1 bitnami/tomcat:latest
```

Figura 43. Instrucción de enlace 2.

La instrucción **–link** seguida del nombre del contenedor que contiene la base de datos permite que la comunicación entre contenedores sea posible, de esta forma se puede mencionar que docker eventualmente será adoptado por muchas empresas ya que ofrece muchas ventajas al momento de crear infraestructuras virtuales.

Al finalizar el enlace no será notorio, pero, docker ofrece una instrucción para mostrar los enlaces entre contenedores.

```
$ docker inspect -f "{{ .HostConfig.Links }}" web

[/db:/web/db]
```

*Figura 44.*Instrucción de enlace 3.

Al ejecutar la instrucción se puede conocer en base a el nombre del contenedor los enlaces que posee.

3.1.7 Docker Hub.

*Figura 45.*Imagen de Docker Hub.

El concepto de docker hub inicia con la necesidad de registrar en la nube imágenes de docker dentro de un repositorio, con la tendencia de migrar imágenes docker que contengan sus propias configuraciones empresariales o personales.

Para hacerlo se debe registrar un usuario en la página https://hub.docker.com/ utilizando un correo electrónico; en este caso se ha ingresado un correo institucional de la Universidad Técnica del Norte.

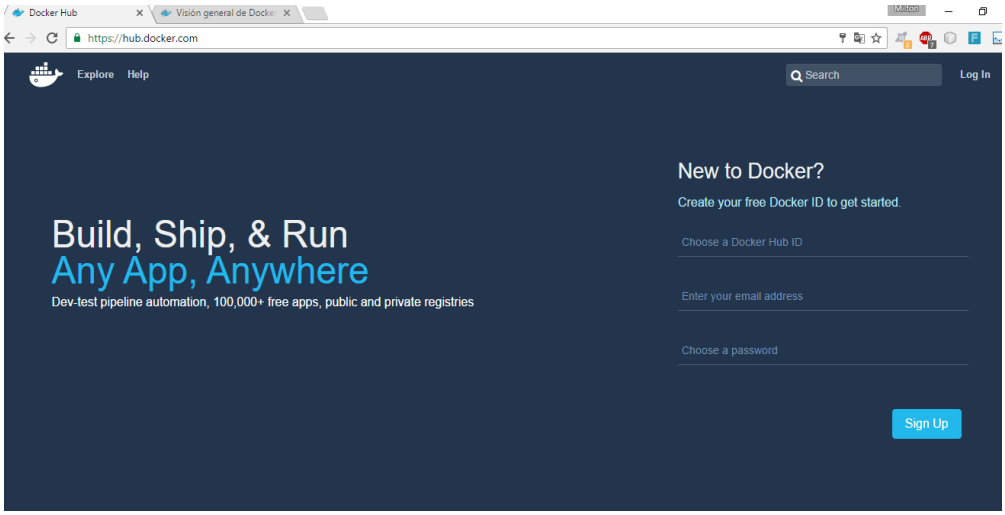


*Figura 46.*Página oficial de docker hub.

Luego de haber llenado los parámetros y confirmado la activación de la cuenta con el correo masimbaniaa@utn.edu.ec requiere la autenticación para ingresar al docker hub.

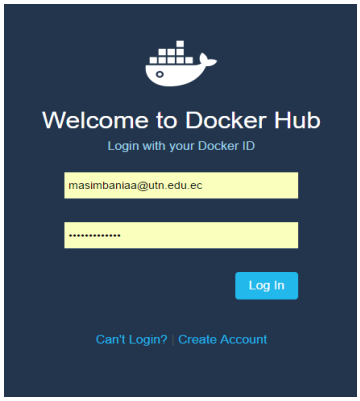


Figura 47. Login docker hub.

Una vez registrados y autenticados presentará la siguiente ventana donde se pueden realizar varias tareas como crear un repositorio, buscar imágenes de docker existentes construidas por otros usuarios de docker, también muestra una lista de todas las instrucciones de Git **push y pull** que se han realizado en las imágenes del repositorio creado.

En este caso se construyeron varias imágenes docker en este repositorio denominado dockerimagentesis, el cual por cada imagen subida se obtienen sus detalles como el peso y las descargas realizadas por los usuarios, esto hace énfasis de que si se publica una imagen docker en un repositorio de docker hub será visible para otros usuarios y podrán obtenerla solo con el nombre anticipando la instrucción Git **pull**.

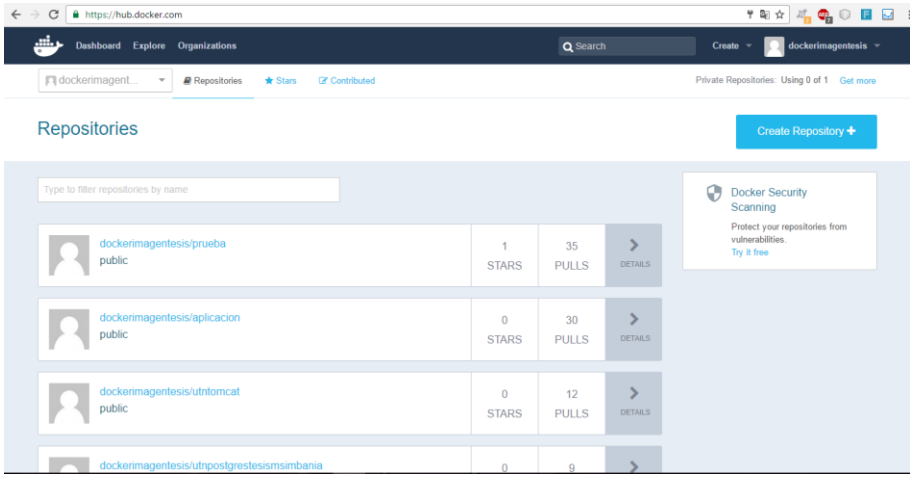


Figura 48. Repositorio docker hub 1.

Para crear un repositorio en docker hub se debe hacer clic en la opción crear repositorio que se encuentra en la parte superior derecha.

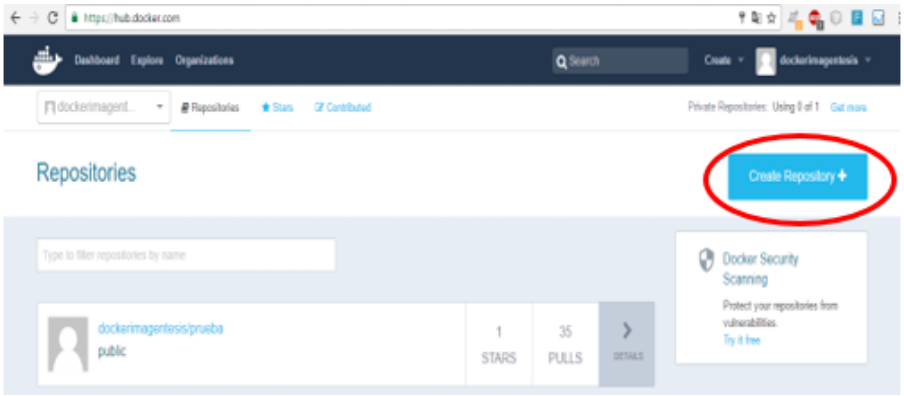


Figura 49. Repositorio docker hub 2.

Después de haber ingresado se muestra la siguiente ventana donde se ingresan los parámetros de reconocimiento de la imagen que se desea realizar la instrucción Git push o subida de la imagen docker que tenemos localmente.

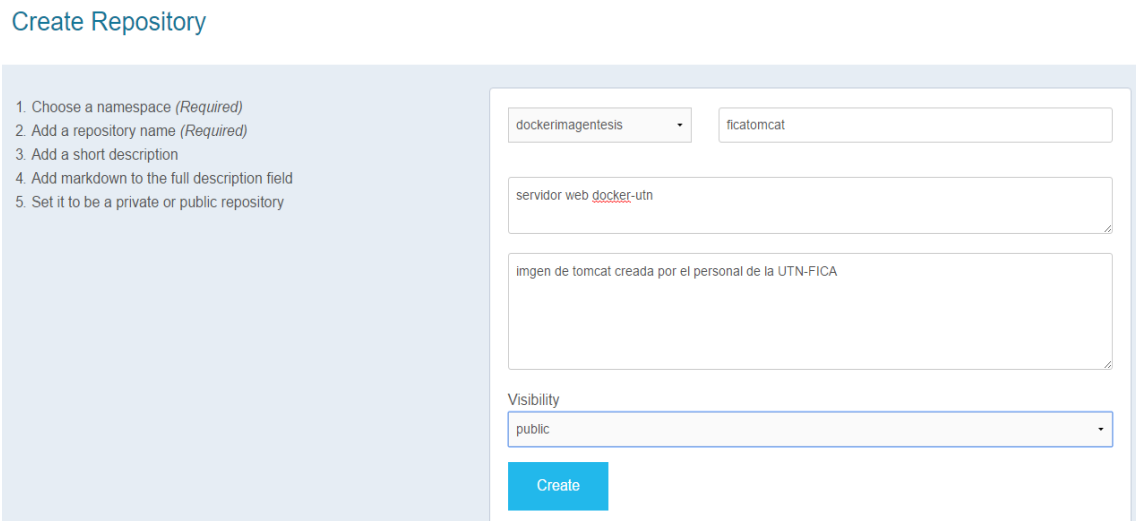


Figura 50. Repositorio docker hub 3.

Al establecer los parámetros de construcción de la imagen en el repositorio de docker hub también se determina si es pública o privada en este caso será publica de este modo todos los usuarios de docker hub podrán visualizar la imagen y proceder a realizar la instrucción de Git **pull** respectiva. Luego se da clic en crear y está listo para poder subir la imagen al repositorio. Construyendo de esta manera un recipiente que contendrá eventualmente la imagen de docker, la pregunta es cómo realizar este procedimiento, y la respuesta es sencilla usando las instrucciones de Git push y pull.

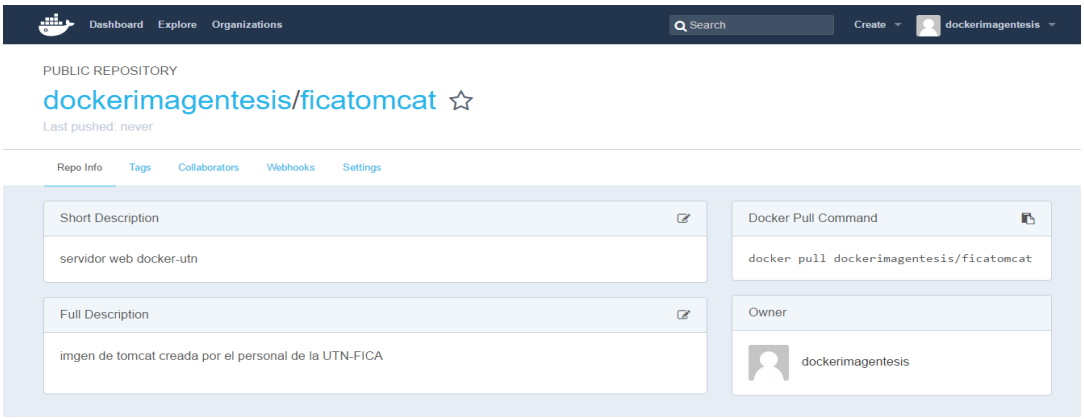


Figura 51. Repositorio docker hub 4.

Para realizar este proceso se debe ejecutar el servidor en Ubuntu que previamente fue instalado docker.

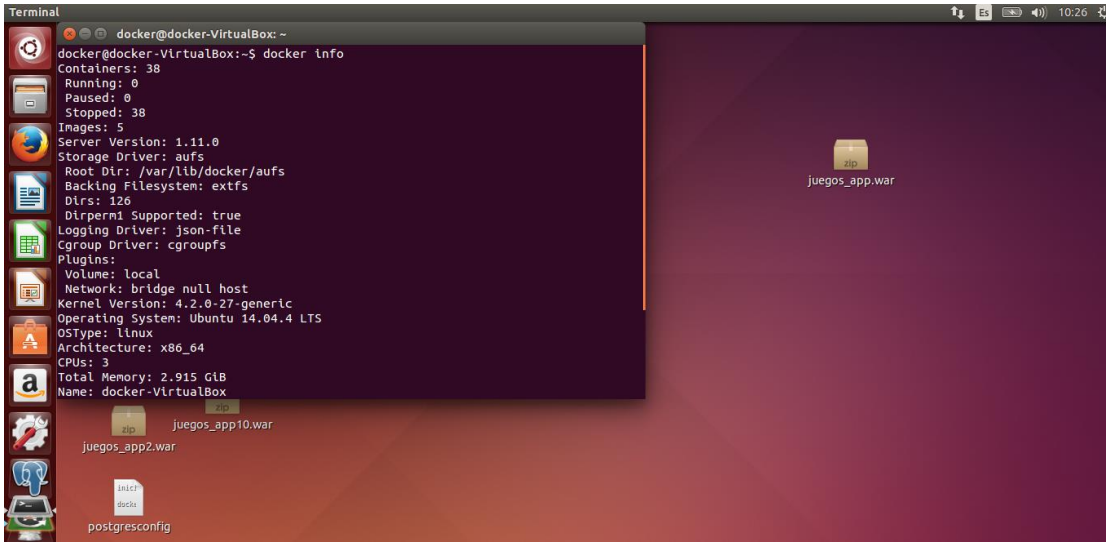


Figura 52. Ubuntu repositorio docker hub 1.

Después de tener inicializado los servicios de docker en Ubuntu 14.04 se puede visualizar las imágenes de docker existentes con el comando **docker images**.

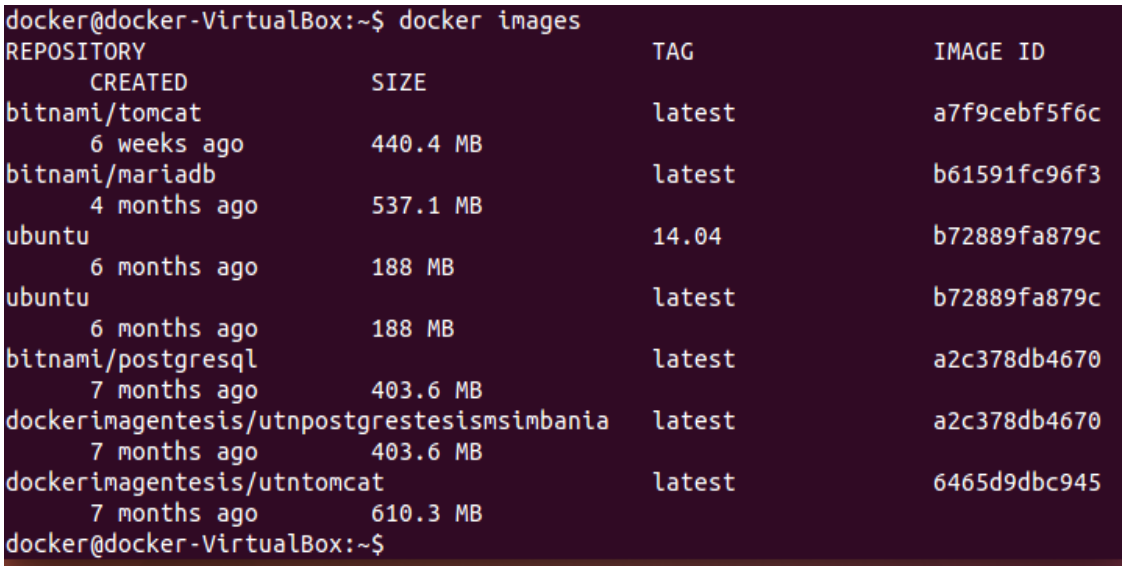


Figura 53. Ubuntu repositorio docker hub 2.

Al ejecutar dicha instrucción como se mencionó anteriormente en este documento se mostrará el tamaño de la imagen, el tag, el id de la imagen seguido del nombre y adicionalmente la fecha de la última iteración de la imgen de docker.

Para lograr introducir las instrucciones de Git push y pull se deben realizar algunos pasos en este caso se hace referencia al tag de la imagen de Tomcat que desea subir al repositorio de docker hub, antes de realizar este proceso debemos tener bien definido los siguientes parámetros:

Tabla 5 *Credenciales Docker hub.*

Referencia	Credencial
Nombre de usuario	Dockerimagentesis
Nombre Repositorio	Ficatomcat
Email	masimbaniaa@utn.edu.ec

Una vez definido estas credenciales se puede realizar una referencia de la imagen de docker al repositorio con un nombre de usuario de la siguiente forma:

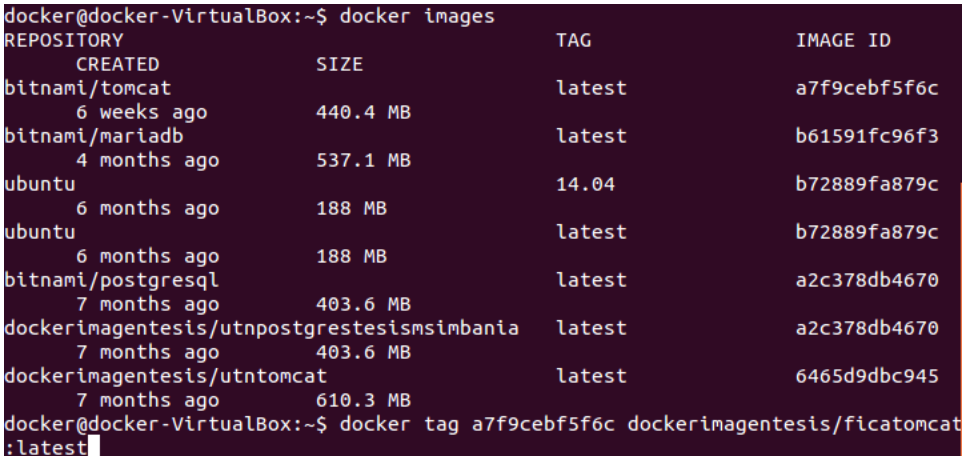


*Figura 54.*Ubuntu repositorio docker hub 3.

Al ejecutarlo no se aprecia mayor cambio porque solo realiza una referencia de nombre hacia un repositorio existente en docker hub.

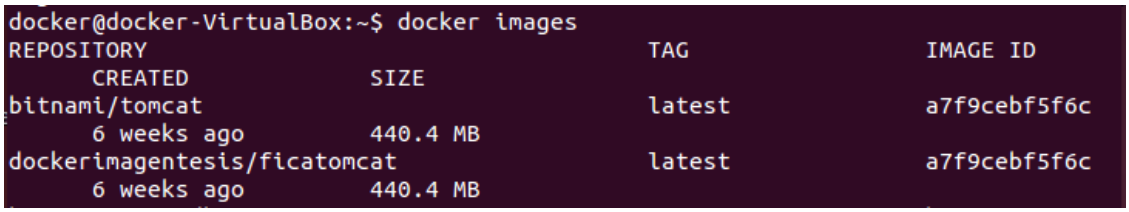


Figura 55. Ubuntu repositorio docker hub 4.

El siguiente paso para lanzar la imagen al repositorio virtual es realizar una autenticación en docker usando la instrucción **docker login.**

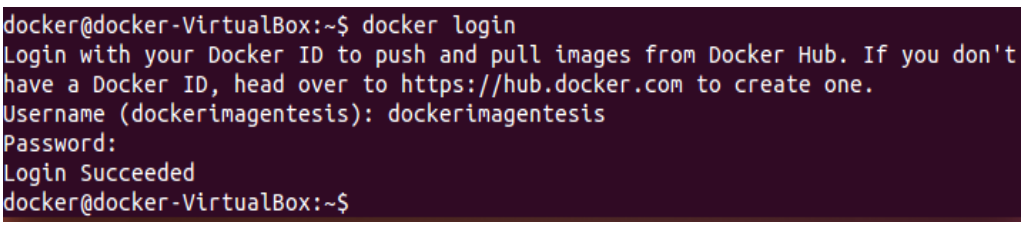


*Figura 56.*Ubuntu repositorio docker hub 5.

Luego se debe ejecutar la siguiente instrucción:

docker push dockerimagentesis/ficatomcat.

```
docker@docker-VirtualBox:~$ docker push dockerimagentesis/ficatomcat
The push refers to a repository [docker.io/dockerimagentesis/ficatomcat]
3ba9837dae55: Mounted from bitnami/tomcat
4c7d7b1c6618: Mounted from bitnami/tomcat
de6d5bc55d44: Mounted from bitnami/tomcat
ce0d592141a4: Mounted from bitnami/tomcat
913181f74166: Mounted from bitnami/tomcat
b9da96731018: Mounted from bitnami/tomcat
b58c930d80c4: Mounted from bitnami/tomcat
9f5c074559e3: Mounted from bitnami/tomcat
ffb6ddc7582a: Mounted from bitnami/tomcat
344f56a35ff9: Mounted from bitnami/tomcat
530d731d21e1: Mounted from bitnami/tomcat
24fe29584c04: Mounted from bitnami/tomcat
102fca64f924: Mounted from bitnami/tomcat
latest: digest: sha256:3e98c49342382b9b00a02cd17f2b4c173cc45146cd3f1eadaecb60b5c
c579858 size: 3015
docker@docker-VirtualBox:~$
```

Figura 57. Ubuntu repositorio docker hub 6.

Finalmente se tiene subida la imagen en el repositorio docker hub. El beneficio de este procedimiento es poder realizar migraciones de contenedores ya configurados y simplemente realizar instrucciones de Git para subirlas u obtenerlas.

De este modo la imagen de Tomcat que se alojó en el repositorio de docker hub se mostrará de esta forma y estará listo para realizar las instrucciones de Git push y pull.

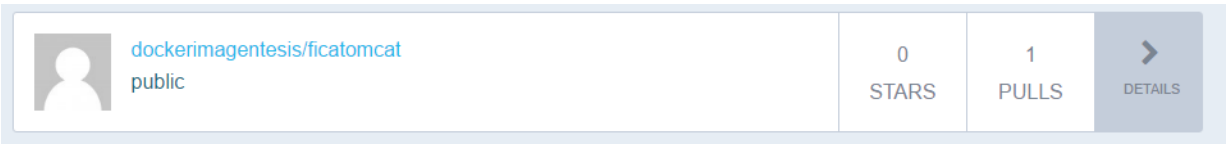


*Figura 58.*Git repositorio docker hub 1.

La instrucción de pull de la imagen del repositorio se encuentra en detalles en la parte superior derecha.

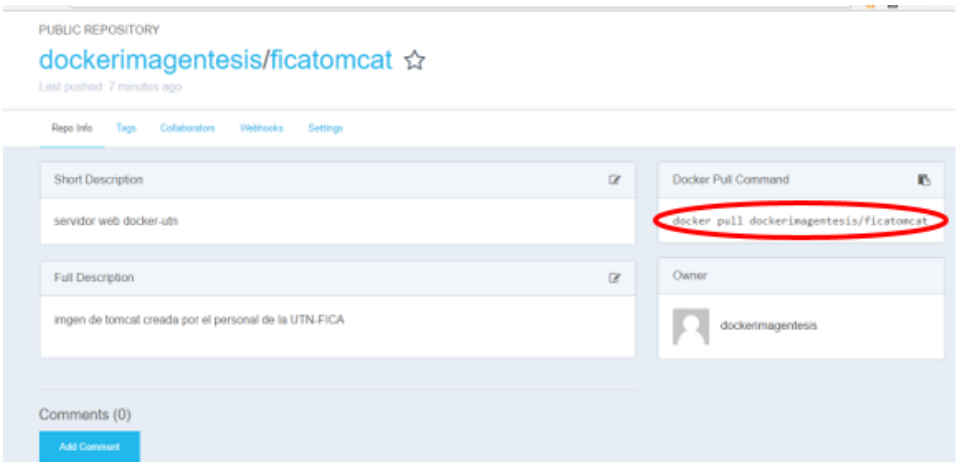


*Figura 59.*Git repositorio docker hub 2.

De cierto modo esto no termina aquí ya que muchos usuarios están construyendo varias imágenes en sus propios repositorios; además la mayoría de ellos tienden a compartir sus imágenes docker haciendo del uso de esta herramienta más sencilla y con mucho soporte entre la comunidad de docker hub.

Para comenzar a hacer búsquedas de imágenes en la comunidad de usuarios de docker hub se debe realizar lo siguiente:

- Establecer qué imagen se desea obtener, en este caso se requiere de una imagen de postgres en la versión 9.2 y para ello se usó la opción de búsqueda ubicada en la parte superior de la página y se debe ingresar el nombre de la imagen que se desea adquirir.

*Figura 60.*Git repositorio docker hub 3.

- Luego de establecer la búsqueda se muestra una lista de todas las imágenes con ese nombre y versión que se estableció.

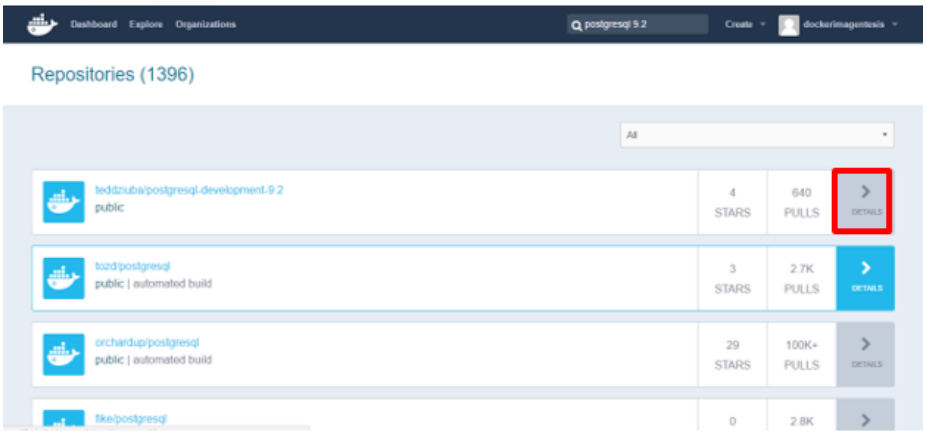


*Figura 61.*Git repositorio docker hub 4.

- Para ver la información del contenedor como sus credenciales y otros parámetros se debe ingresar a la opción detalles que despliega toda la información acerca de la imagen docker.

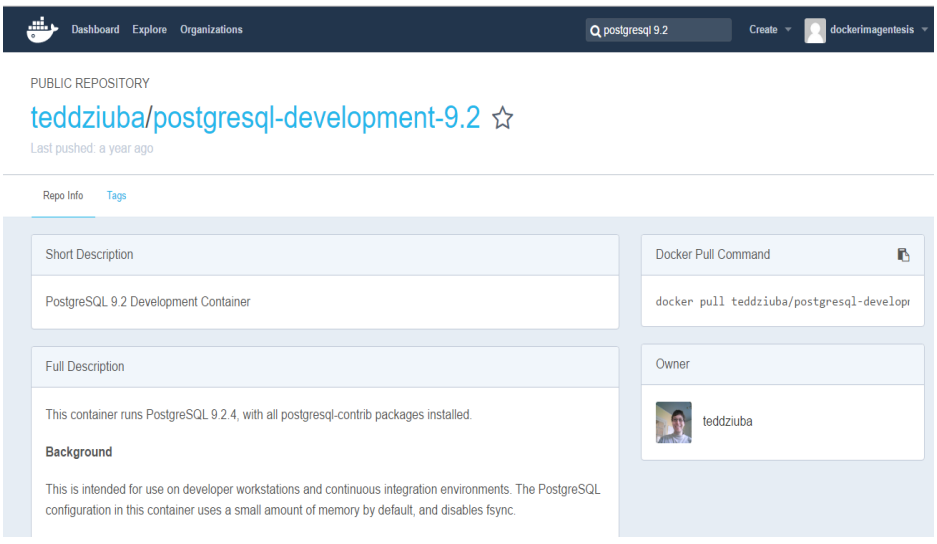


*Figura 62.*Git repositorio docker hub 5.

3.1.8 Actualización y desinstalación de los componentes Docker.

Figura 63. Actualizaciones Docker.

Actualizaciones

Parte de docker y además una ventaja es la facilidad de las actualizaciones que brinda la herramienta en sí como las imágenes que contenga.

Para actualizar la herramienta docker se debe ejecutar la misma instrucción de instalación y para las imágenes se seguiría el mismo proceso.

Para obtener las actualizaciones se deberá ejecutar la misma instrucción de Git pull de la imagen docker, por ejemplo: se encuentra instalado en un servidor docker una base de datos PostgresSQL con la versión 9.2 y se desea actualizar a una versión actual que será la versión 9.6 la respuesta para la actualización de la imagen docker será ejecutar la misma instrucción de Git pull para PostgresSQL.

```
docker pull bitnami/postgresql:latest
```

*Figura 64.*Git pull de PostgresSQL.

Desinstalaciones

Para desinstalar la herramienta docker se deberá ejecutar las siguientes instrucciones que harán que se elimine todos sus archivos.

```
$ sudo apt-get purge docker-engine
```

*Figura 65.*Desinstalación docker parte 1.

```
$ sudo apt-get autoremove
```

*Figura 66.*Desinstalación docker parte 2.

```
$ rm -rf /var/lib/docker
```

Figura 67. Desinstalación docker parte 3.

Para eliminar las imágenes creadas por propia autoría o las proporcionadas por Bitnami se deberá realizar solamente una instrucción seguida de la ID de la imagen.

Ejemplo

```
$ docker rmi -f 7d9495d03763
```

Figura 68. Eliminación de una imagen docker.

3.2 Herramienta Bitnami

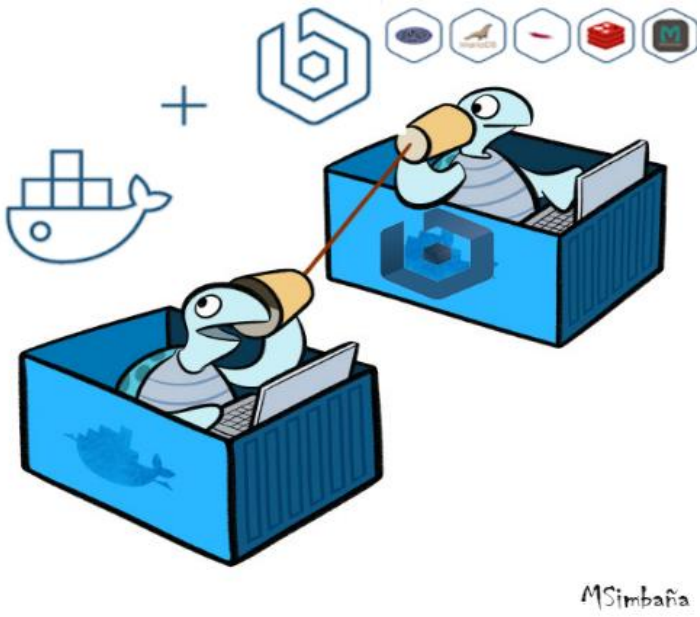


Figura 69. Comunicación Docker y Bitnami

Bitnami es una herramienta que ofrece múltiples instaladores, recientemente Bitnami aportó con Docker al crear imágenes Bitnami que contengan instalaciones de herramientas necesarias para elaborar aplicaciones es por eso que en este estudio se usan imágenes Bitnami en Docker con la finalidad de tener instalaciones actualizadas y bien estructuradas.

Para obtener las imágenes de Bitnami compatible con docker ingresamos a la página web https://bitnami.com

Después de ingresar a la página https://bitnami.com/containers procedemos a registrar un usuario y contraseña, el motivo de autenticarnos es el de tener la visibilidad de todas las imágenes que brinda Bitnami al desarrollador que usa docker.

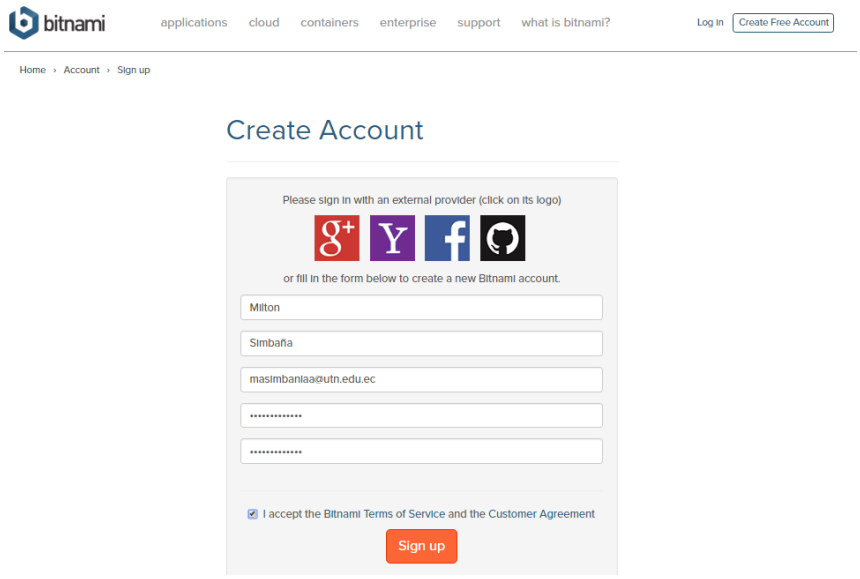


*Figura 70.*Cuenta de Bitnami parte 1.

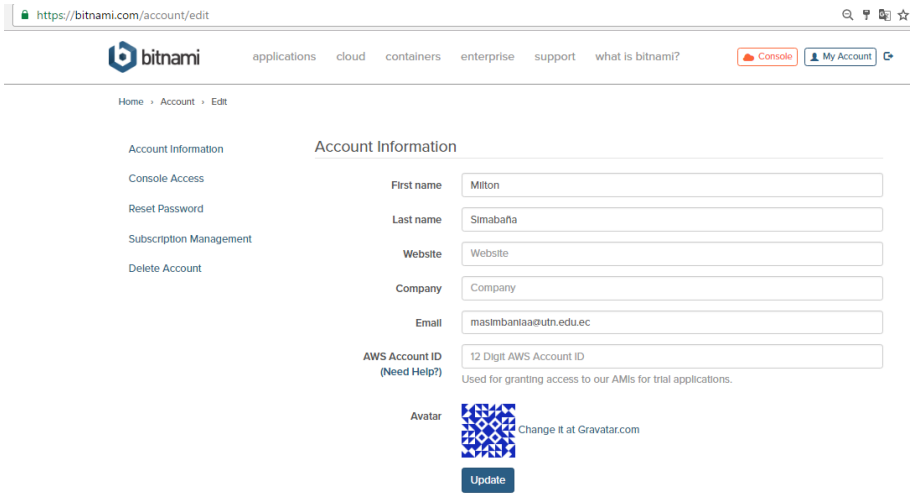


Figura 71. Cuenta de Bitnami parte 2.

Después de haber completado el registro en Bitnami procedemos a buscar las imágenes docker disponibles.

Bitnami brinda una diversidad de imágenes compatibles con docker por ello esta herramienta se vuelve aún más atractiva para los desarrolladores y administradores ya que tenemos la

oportunidad de crear infraestructuras virtuales según lo necesitemos de una manera rápida, fácil y sobretodo con un tamaño mínimo.

Bitnami pone a disposición y uso las siguientes imágenes compatibles con docker.

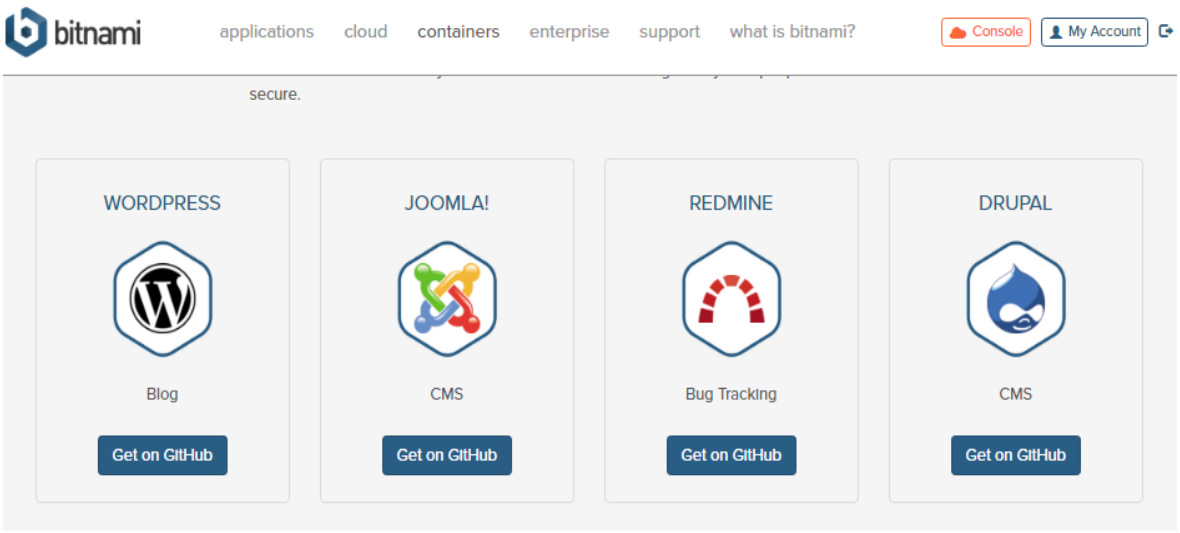


Figura 72. Imágenes Bitnami Docker 1.

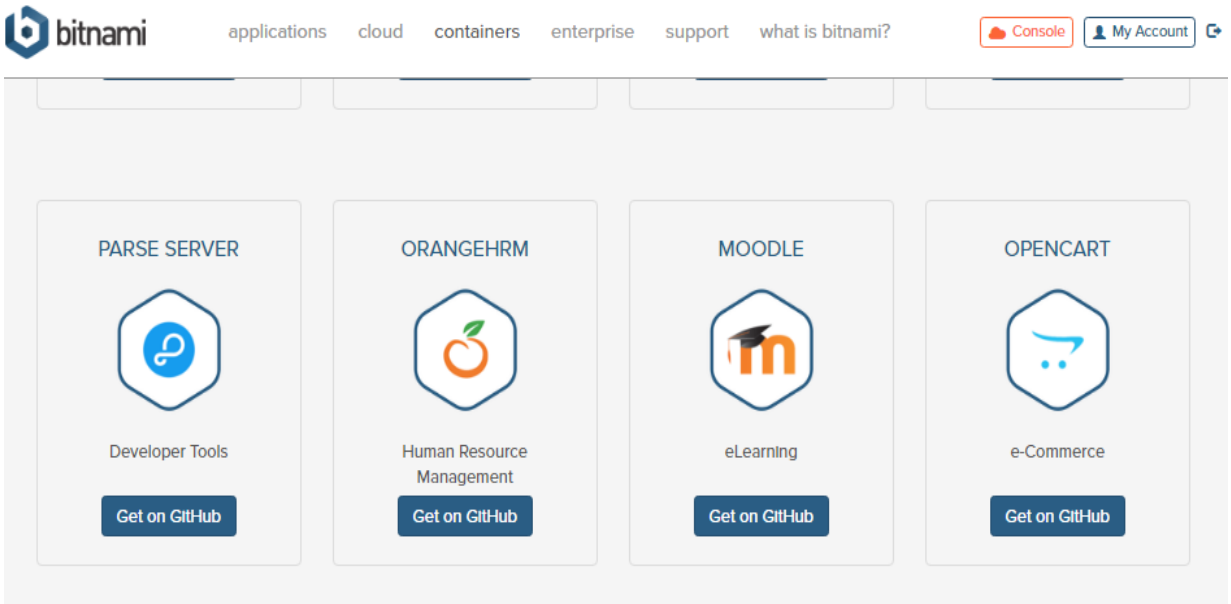


Figura 73. Imágenes Bitnami Docker 2.

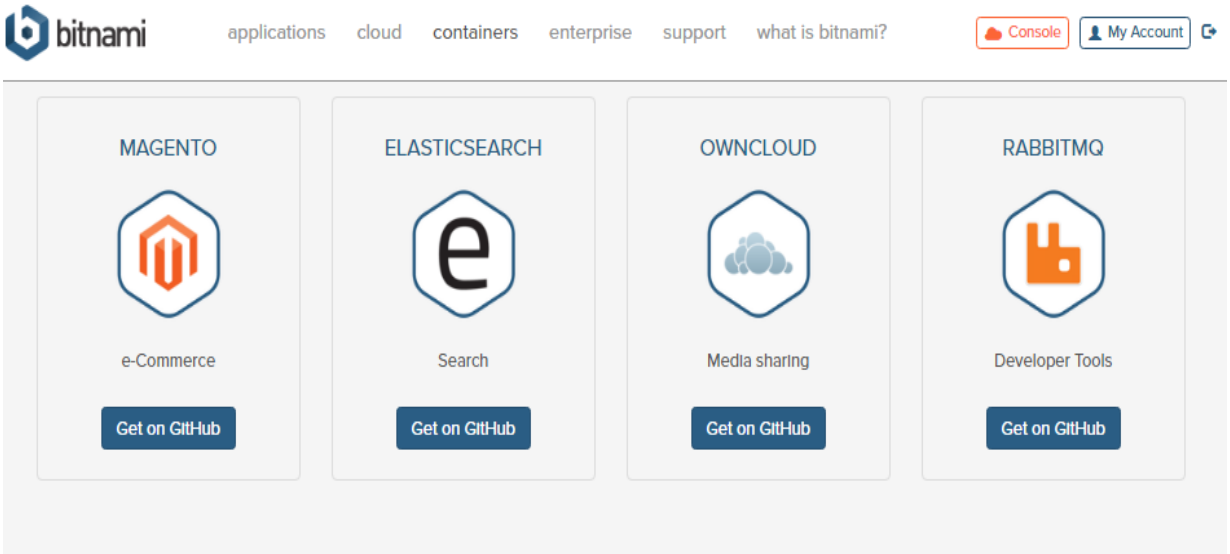


Figura 74. Imágenes Bitnami Docker 3.

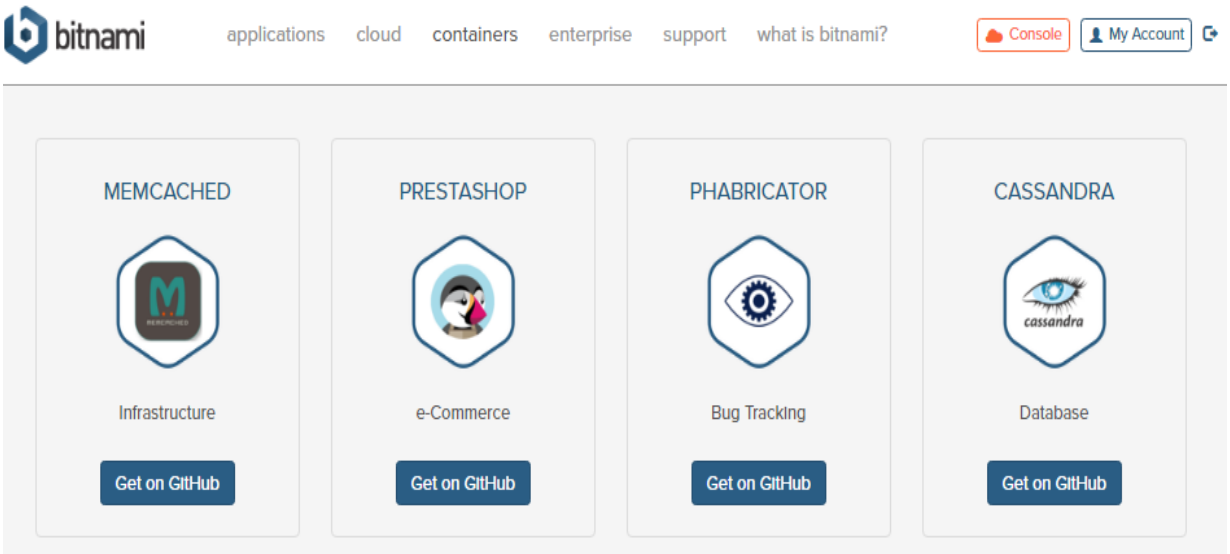


Figura 75. Imágenes Bitnami Docker 4.

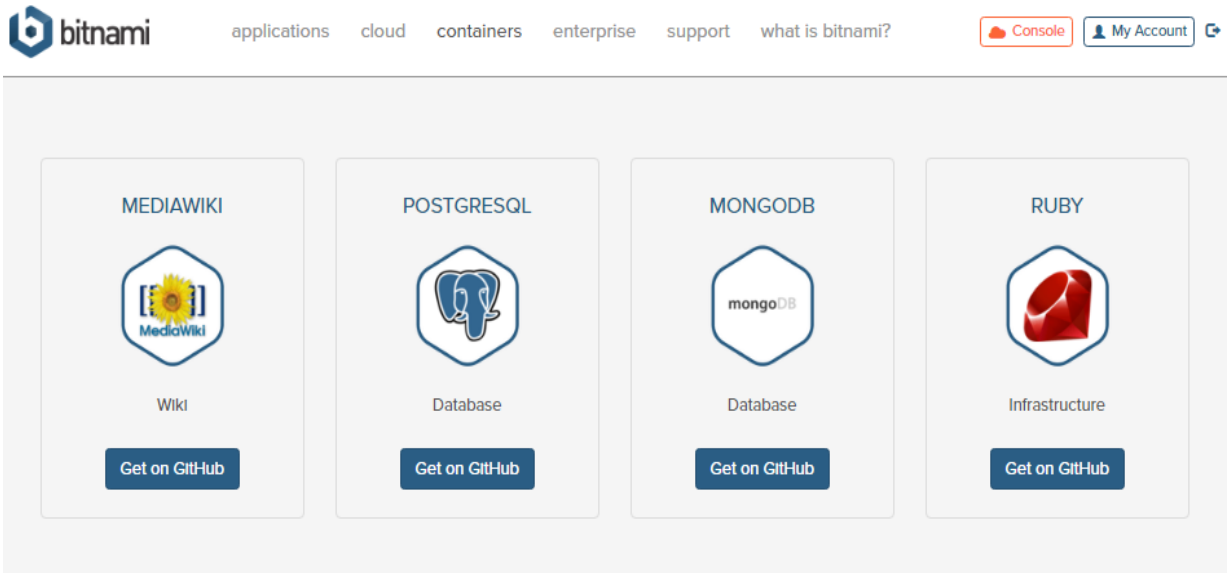


Figura 76. Imágenes Bitnami Docker 5.

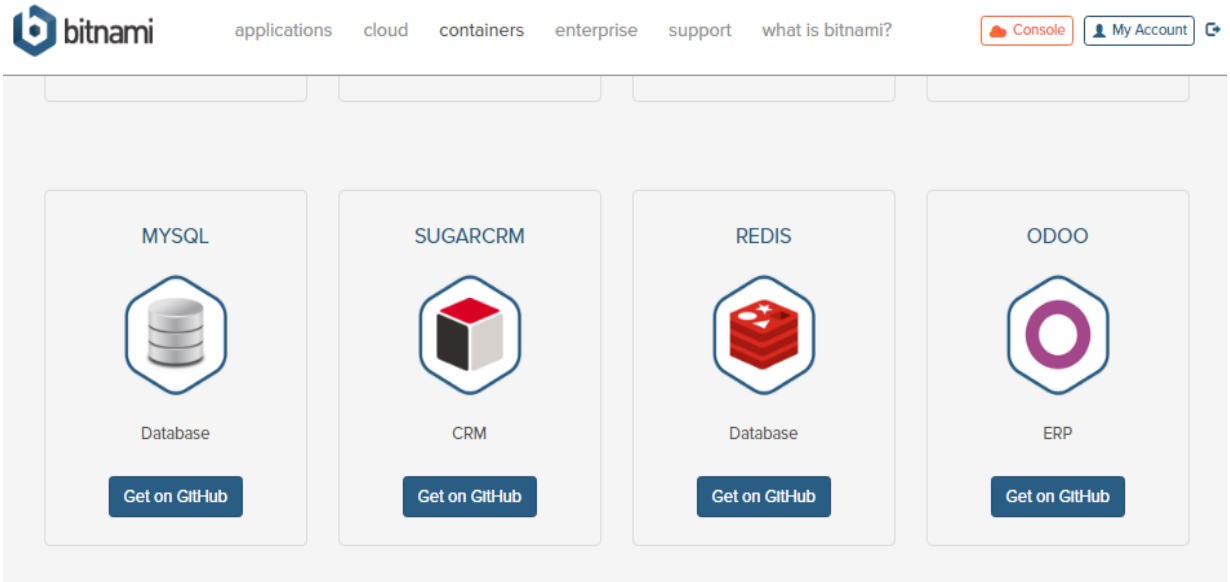


Figura 77. Imágenes Bitnami Docker 6.

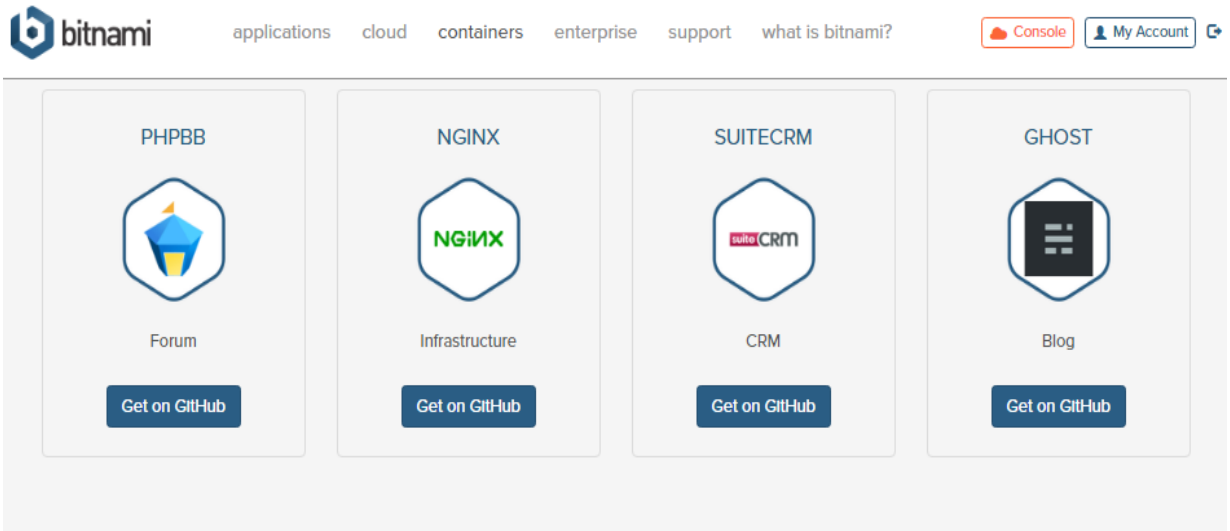


Figura 78. Imágenes Bitnami Docker 8.

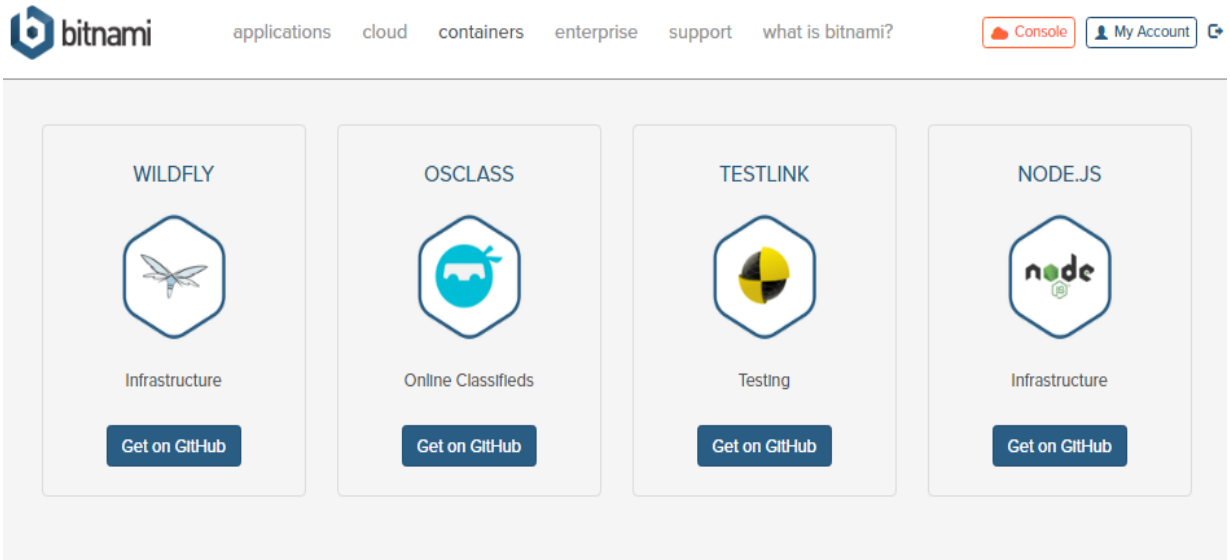


Figura 79. Imágenes Bitnami Docker 9.

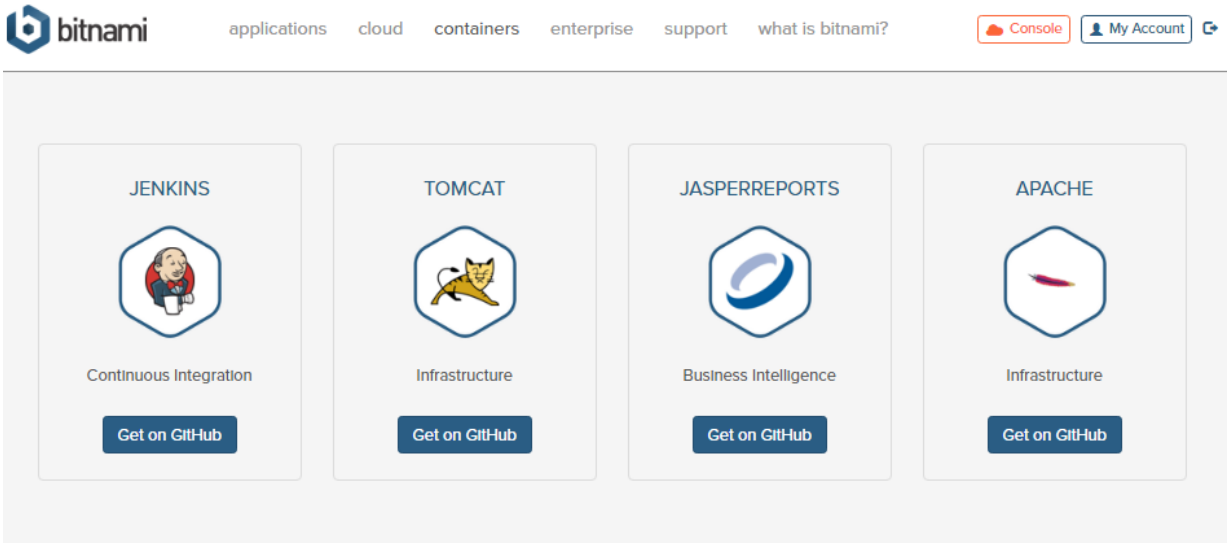


Figura 80. Imágenes Bitnami Docker 10.

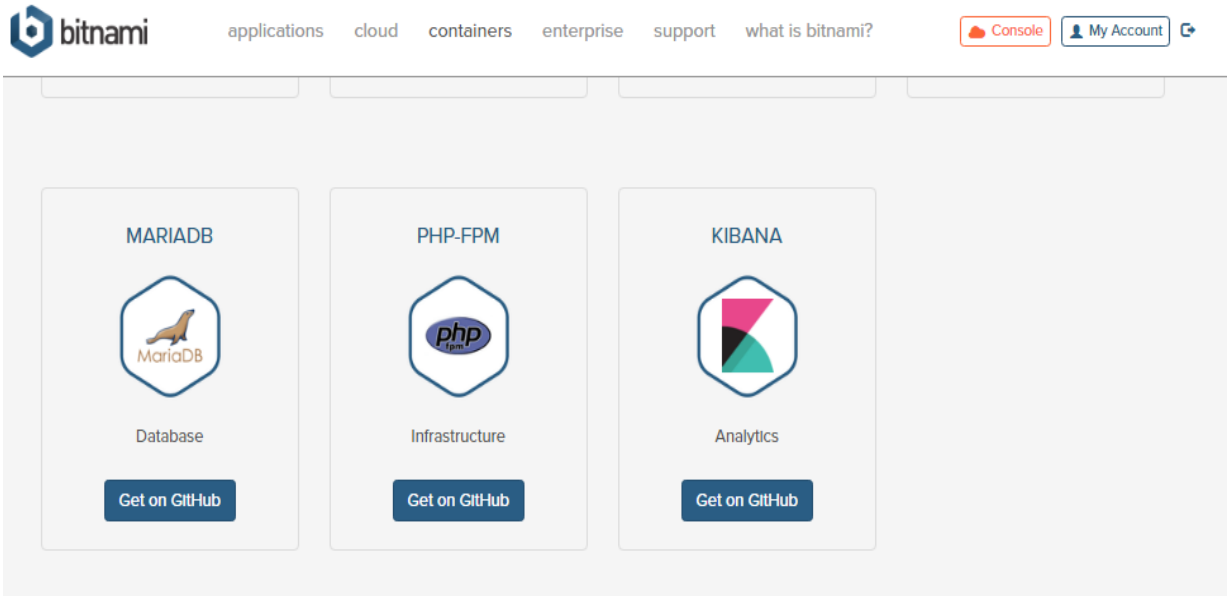


Figura 81. Imágenes Bitnami Docker 11.

3.2.1 Adquirir imágenes Bitnami compatibles con Docker.

Figura 82. Bitnami mas Docker.

Todas las imágenes Bitnami se obtienen en GitHub usando la sentencia docker pull,,este realiza una petición desde el sistema operativo con el fin de que descargue la imagen solicitada al servidor local.

Ejemplo:

Si se desea obtener la imagen Bitnami de Tomcat en Docker dirigirse a la página web: https://bitnami.com/docker

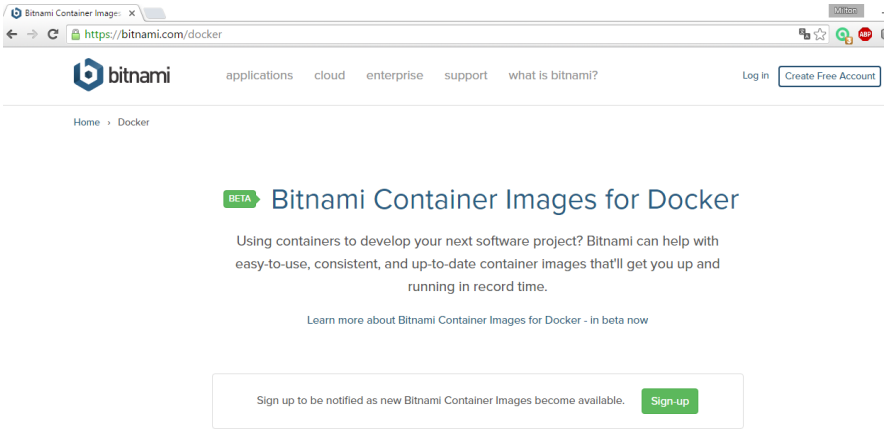


Figura 83. Página oficial Bitnami Docker.

Se selecciona la imagen deseada en este caso Tomcat y presionando la opción Get on GitHub para obtener las características de la imagen y la instrucción de Git para obtenerla.

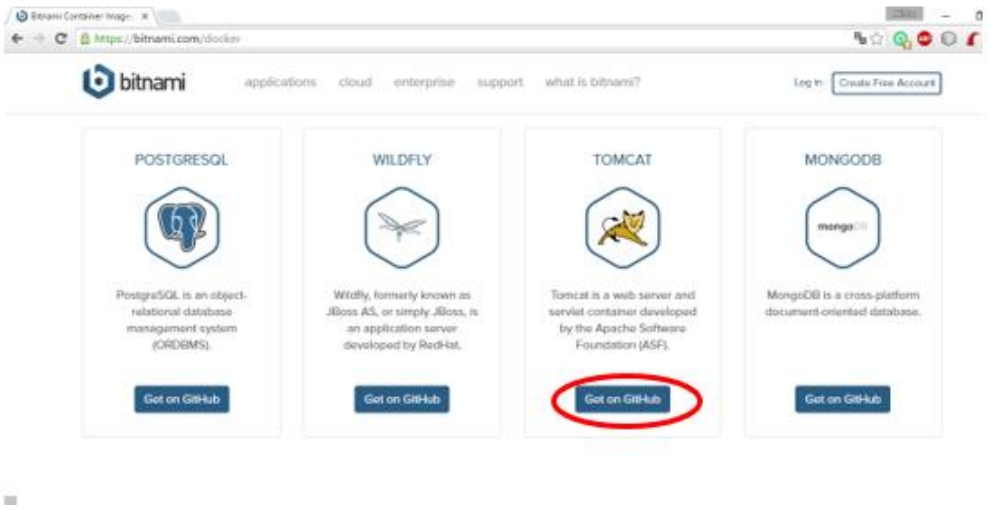


Figura 84. Imagen Bitnami Docker Tomcat.

Se desplegará la siguiente información de la imagen detallando la estructura de su instalación en el Dockerfile.

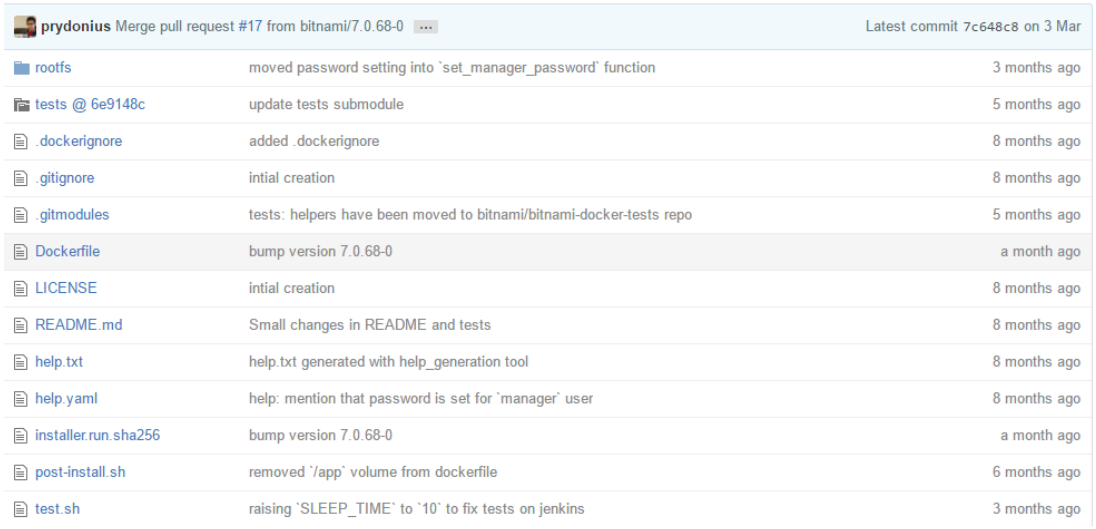


Figura 85. Contenido de la imagen Bitnami Docker Tomcat.

En la parte inferior se desplegará información acerca de la imagen Bitnami/Tomcat y muestra cómo obtenerla, además indica ciertas configuraciones adicionales que serán relevantes para el desarrollador.

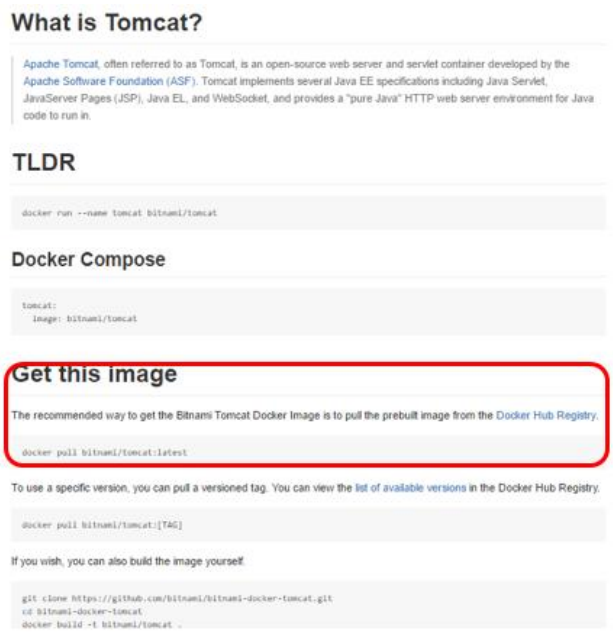


Figura 86. Instrucción para obtener la imagen Bitnami Docker Tomcat.

Para obtener la imagen de Bitnami se debe ejecutar el terminal de Ubuntu y Copiar la instrucción de Git pull para obtener la imagen en este caso será la siguiente:

docker pull bitnami/tomcat:latest

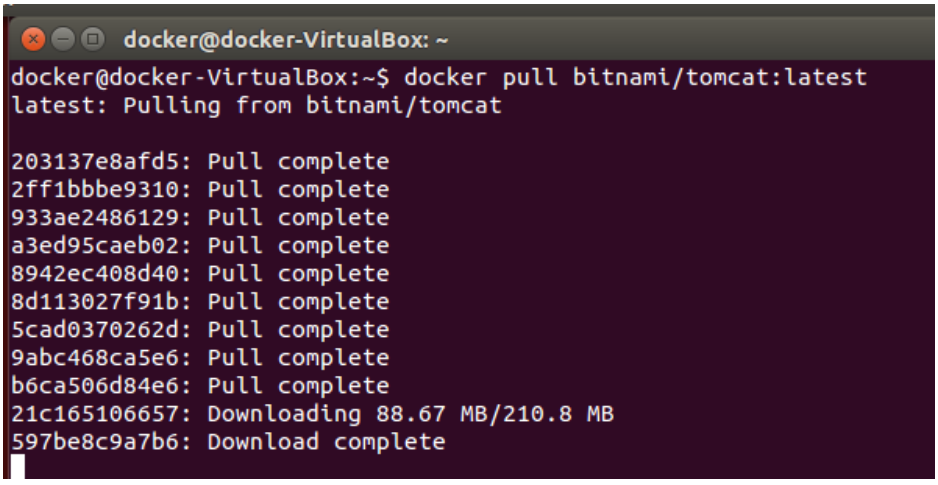


Figura 87. Ejecución de la instrucción para obtener la imagen Tomcat.

Para el desarrollo de esta tesis se realizó la instrucción de Git pull de dos imágenes de Bitnami que ayudaron en la comprensión de esta herramienta además de construir con ellas una infraestructura virtual.

Las imágenes obtenidas desde Bitnami fueron la imagen de Tomcat en la versión 8.0.36 y finalmente la imagen de PostgresSQL en la versión 9.6.

Figura 88. Uso de Tomcat y postgres dentro de docker.

CAPITULO 4

Desarrollo de una aplicación de prueba

4.1 Introducción

Docker permite crear infraestructuras virtualizadas de una manera aislada del sistema anfitrión, para verificar la funcionalidad de la herramienta se realizó el despliegue de una aplicación en Java Server Faces (JSF) que prácticamente es un Framework web que brinda un mejor diseño y estructura en programación web, este tipo de Framework's se usa tradicionalmente para realizar aplicaciones.

De esta forma se realizó una aplicación llamada juegos deportivos el cual tiene una arquitectura cliente servidor que comúnmente se usa en diferentes aplicaciones empresariales Java Enterprise Edición (JavaEE) (Martínez, Aranda, & López, 2010).

En esta tecnología se realizó una infraestructura basada en contenedores que dentro de sí tienen una instalación independiente necesaria para la construcción de una infraestructura virtual, al crear los contenedores esta herramienta posibilita la activación, desactivación e incluso la eliminación de un contenedor con el fin de ejecutar solamente lo necesario para alojar aplicaciones.

Estructuralmente se ha creado dos contenedores el cual uno de ellos contiene la base de datos y otro contiene el servidor web en este caso sería PostgresSQL y Tomcat estas herramientas posibilitan el despliegue de la aplicación en JSF.

Tomcat o apache Tomcat en realidad funciona como un repositorio de Servlets, este servidor web trata íntimamente con la máquina virtual de java (VMJ) muchos desarrolladores sostienen que Tomcat es el servidor web que más se utiliza actualmente cuando se trabaja con java en entornos web. (Gómez Jiménez, 2012)

Estos dos contenedores que contienen las instalaciones de postgres y Tomcat se los obtuvo a partir de un Dockerfile en la página oficial de Bitnami de forma que ayudó en las instalaciones de

las imágenes Docker, las cuales posteriormente se procedió a ejecutarlas y configurarlas con fin de alojar la aplicación JSF desarrollada.

Finalmente se usó una metodología de desarrollo web para analizar y verificar el comportamiento de una aplicación JSF usando Docker como Administrador Virtual.

Figura 89. Estructura de desarrollo Docker.

4.2 Desarrollo de la aplicación usando la metodología SCRUM

4.2.1 Plan de desarrollo del proyecto.

<Desarrollo de un sistema web para automatizar el registro de inscripciones en eventos deportivos.>
Plan de Desarrollo del Proyecto

Versión 0.1

Historial de Revisiones

Tabla 6 *Historial de revisiones plan de proyecto(SCRUM).*

Fecha	Versión	Descripción	Autor
25/04/2016	0.1	Versión preliminar como propuesta de desarrollo	Milton Andrés Simbaña Alarcón
02/05/2016	0.2	Versión preliminar como propuesta de desarrollo	Milton Andrés Simbaña Alarcón
09/05/2016	0.3	Versión preliminar como propuesta de desarrollo	Milton Andrés Simbaña Alarcón
16/05/2016	0.4	Versión preliminar como propuesta de desarrollo	Milton Andrés Simbaña Alarcón
23/05/2016	0.5	Versión preliminar como propuesta de desarrollo	Milton Andrés Simbaña Alarcón

Plan de desarrollo del proyecto

Introducción.

Este plan de proyecto del desarrollo e investigación "Desarrollo de un sistema web para automatizar el registro de inscripciones en eventos deportivos" es una versión preliminar preparada para ser tomada en cuenta como una propuesta de implementación a eventos deportivos. En esta parte del documento se proveerá una visión global del desarrollo de la aplicación.

El proyecto ha sido considerado para fortalecer el proceso de inscripciones en juegos deportivos, con la finalidad de tener una automatización al realizar eventos del mismo, a través del desarrollo de una aplicación, usando la metodología SCRUM.

El enfoque de desarrollo propuesto constituye el desarrollo de una aplicación en Java Server Faces (JSF), trabajando conjuntamente con la base de datos Postgres, de tal forma que se definirá las características del proyecto, seleccionando los roles de los participantes, las actividades a realizar y los artefactos (entregables) que serán subministrados en su respectivo momento.

Propósito.

El propósito de este plan de desarrollo del proyecto es proporcionar información necesaria para obtener una idea global el proyecto. En él se describe el enfoque de desarrollo de la aplicación JSF.

Los usuarios del plan de proyecto son:

- El jefe del proyecto el cual controla, organiza la agenda y realiza un seguimiento sigiloso del proyecto.
- Los miembros del equipo de desarrollo lo usan para entender lo que se debe hacer cuando deben hacerlo y que otras actividades dependen de ello.

Alcance.

En este plan de desarrollo de proyecto se identificará el propósito, alcance y los objetivos del proyecto de igual forma se definirá los roles del equipo de trabajo que formarán parte del desarrollo del proyecto, de esta manera se podrá facilitar la visión del jefe o líder del proyecto.

Resumen.

El plan de proyecto está organizado en las siguientes secciones:

- **Vista general del proyecto. -** proporciona una descripción del propósito, alcance y objetivos del proyecto, estableciendo los artefactos que serán producidos y utilizados durante el proyecto.
- **Organización del proyecto. –** describe la estructura organizacional del equipo de desarrollo.
- **Gestión del proceso.** –explica costos y planificación estimada del proyecto, además de realizarse un seguimiento.
- **Planes y guías de aplicación.** - provee información global del desarrollo del software a realizarse incluyendo métodos, herramientas utilizadas.

Vista General del Proyecto.

Objetivos del proyecto:

- Determinar el proceso que se debe realizar para la inscripción en juegos deportivos.
- Realizar la gestión de roles que controlen las actividades de los usuarios.
- Crear eventos basados en fechas para permitir a un usuario inscribirse en una de ellas.
- Diseñar y desarrollar una aplicación que permita agilizar estos procesos con el fin de registrar las inscripciones con sus respectivos equipos y jugadores.

Suposiciones y restricciones.

Suposiciones.

- El jefe del proyecto se responsabiliza en asignar Usuarios Funcionales con conocimientos y experiencia en los temas asociados del aplicativo, también se encargará de asignar usuarios técnicos que participen en el proceso de pruebas y aceptación del sistema.
- Cualquier requerimiento o actividad adicional que no esté contemplada en este plan de proyecto estará fuera del alcance del proyecto.

Restricciones.

- El proyecto tiene un presupuesto y tiempo estimado para su finalización (2 meses, tres mil dólares).
- Para desplegar la aplicación se deberá contar con un servidor local que tenga instalado Docker que permita alojar la aplicación.

Entregables del proyecto.

En esta sección se describe cada uno de los artefactos que serán generados y utilizados por el proyecto es decir los entregables.

Es preciso mencionar que de acuerdo con la metodología SCRUM para el sistema web todos los artefactos son objeto de modificaciones a lo largo del proceso de desarrollo, con lo cual al finalizar este proceso tendremos una versión definitiva y completa de cada uno de ellos, sin embargo, el resultado de cada iteración y los hitos del proyecto están enfocados a conseguir un cierto grado de completitud y estabilidad de los artefactos.

- **Sprint Planning Meeting** (Reunión de planificación del Sprint): Esta reunión se realiza cada 15 o 30 días, en esta sección incluye el análisis de plan de proyecto y requerimientos 830 en esta reunión se pretende:

- ✓ Seleccionar que trabajo de hará.
- ✓ Prepara con el equipo completo el Sprint Backlog que detalla el tiempo llevara hacer el trabajo.
- ✓ Organizar y definir un alcance favorable que se realice durante el Sprint.
- ✓ Esta planificación tiene un límite máximo de 8 horas.

- **Daily Scrum:** Se refiere a las reuniones diarias para analizar el estado de un proyecto con el que se debe cumplir ciertas normas:

- ✓ Las reuniones inician en horas puntuales diariamente.
- ✓ Todos son bienvenidos, pero solo las personas involucradas en el proyecto pueden opinar acerca del proyecto.
- ✓ La reunión diaria tiene una hora fija de 15 minutos.

- **Sprint Review Meeting:** Son reuniones de revisión de máximo 4 horas de duración donde se tratan asuntos como:

- ✓ Revisar tareas asignadas diariamente definiendo si fue completado o no.
- ✓ En caso de no haber completado la tarea el equipo de desarrollo ayudará a completar la tarea para continuar con la siguiente tarea asignada.
- ✓ Presentar el trabajo completado (Demo)

- **Sprint Retrospective:** Al finalizar cada Sprint **s**e analiza el trabajo ya sea semanal o mensualmente del equipo evaluando el avance del proyecto, es decir en esta sección se pretende la mejora continua del proceso. Esta reunión tiene un tiempo fijo de 4 horas.

Organización del Proyecto.

Participantes en el proyecto:

Tabla 7 *Dirección del proyecto (SCRUM).*

Dirección del proyecto		
Nombre	**Función**	**Nominación**
Xavier Mauricio Rea Peñafiel	Ingeniero en sistemas	Director de proyecto.

Tabla 8 *Participantes del proyecto (SCRUM).*

Participantes en el Proyecto		
Nominación	**Perfil Profesional**	**Nombre**
Jefe de Proyecto	Ingeniero/a en Sistemas Computacionales. Especialista en gestión de procesos. Especialista en desarrollo de proyectos de software. Especialista en procesos académicos dirigido a estudiantes y graduados universitarios.	Xavier Mauricio Rea Peñafiel.
Administrador de Base de Datos	Estudiante de Ingeniería en Sistemas Computacionales. Especialista en desarrollo de proyectos de software. Conocimiento intermedio en bases de datos.	Milton Andrés Simbaña.
Analista de Sistemas	Estudiante de Ingeniería en Sistemas Computacionales. Especialista en desarrollo de software. Programador.	Milton Andrés Simbaña.
Administrador de Docker	Estudiante de Ingeniería en Sistemas Computacionales, Especialista en crear infraestructuras virtuales usando contenedores Docker, administrador de aplicaciones.	Milton Andrés Simbaña.

Interfaces Externas.

El jefe de proyecto definirá los requerimientos del sistema al equipo de desarrollo de tal forma que los miembros del equipo evaluarán los artefactos de acuerdo a cada subsistema y según el plan establecido.

Roles y Responsabilidades.

En esta sección se define las responsabilidades de cada uno de los puestos en el equipo de desarrollo durante las fases de inicio y elaboración de acuerdo a los roles asignados.

Tabla 9 *Puestos y responsabilidades (SCRUM).*

Puesto	Responsabilidad
Jefe de proyecto	Asigna los recursos, gestiona prioridades, coordina las iteraciones con los clientes y usuarios, mantiene al equipo enfocado a los objetivos propuestos ,asegura la integridad y calidad de los artefactos del proyecto, supervisara el desarrollo del proyecto además de analizar su arquitectura y control del mismo.
Administrador de base de datos	Diseña la base de datos según los requerimientos del sistema , y verifica las respectivas pruebas funcionales.
Analista de sistemas	Preparación de las pruebas funcionales según los requerimientos ,elabora la documentación del proyecto además de elaborar modelos de implementación y despliegue de la aplicación.
Administrador de Docker	Crea un ambiente estructural donde se va a desplegar la aplicación, determina accesos a la Base de Datos (BDD) y a el servidor web.

Gestión del Proceso.

Estimaciones del proyecto.

El presupuesto del proyecto y los recursos involucrados se detallan a continuación:

Talento Humano.

Tabla 10 *Presupuesto del proyecto talento humano (SCRUM).*

Descripción	Nro. Horas	Costo Hora	Costo($)
Desarrollador del proyecto (Costo de trabajo por hora)	320	8	2.560.00
		TOTAL	**2.560.00 dólares**

Recursos Material: (Hardware y Movilidad).

Tabla 11 *Presupuesto del proyecto Recursos de material (SCRUM).*

Descripción	Costo($)
Computadora	1000.00
Internet	50.00
Alimentación	200.00
Imprevistos	100.00
TOTAL	**1350.00 dólares**

Capacitación y herramientas tecnológicas.

Tabla 12 *Capacitación y herramientas (SCRUM).*

Descripción	Costos($)
Capacitación en Postgres SQL	500.00
Capacitación de Herramientas Docker	1500.00
Capacitación de Framework JSF	500.00
Capacitación de JAVAEE	500.00
Capacitación de Framework Bootfaces	500.00
TOTAL	**3.500.00 dólares**

Total, de Gastos del Proyecto.

Tabla 13 *Costo total del proyecto de software final (SCRUM).*

Descripción	Costo($)
Talento Humano	2.560.00
Recursos Material: (Hardware y Movilidad)	1.350.00
Capacitación y herramientas tecnológicas	3.500.00
TOTAL	**7410.00 dólares**

Plan de las fases.

El desarrollo se llevará a cabo en base a faces con una o más iteraciones dentro de cada una de ellas, de tal manera que en esta sección se pretende mostrar las distribuciones en tiempos y numero de iteraciones de cada fase.

Tabla 14 *Duración estimada del proyecto en fases de SCRUM.*

Fase	Nro. Iteraciones	Duración
Sprint Planning Meeting	4	80 horas
Daily Scrum	4	80 horas
Sprint Review Meeting	4	80 horas
Sprint Retrospective	4	80 horas
	TOTAL	**320 Horas**

Calendario del proyecto.

A continuación, se muestra un cronograma de las principales tareas que se asignarán al proyecto y adicionalmente se definirá cuando serán generados los artefactos en mayor o menor grado de acuerdo a la fase de iteración del proyecto.

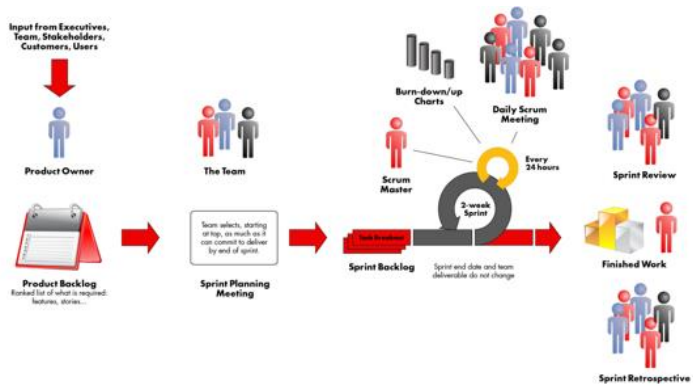


Figura 90. Estructura Metodología SCRUM. (Labs, 2016)

Tabla 15 *Cronograma de actividades del proyecto (SCRUM).*

Disciplinas / Artefactos generados	Comienzo	Aprobación
Fases del desarrollo del proyecto – SCRUM		
Primer Sprint		
Plan de proyecto –Fase de elaboración	23/05/2016	25/05/2016
Requerimiento 830 –Fase de elaboración	Diario durante todo el proyecto	
Informe Arquitectónico – Fase de elaboración	27/05/2016	30/05/2016
Segundo Sprint		
Plan de proyecto –Fase de elaboración	Diario durante todo el proyecto	
Requerimiento 830 –Fase de elaboración	31/05/2016	03/06/2016
Informe Arquitectónico – Fase de elaboración	06/06/2016	07/06/2016
Modelamiento de la Base de datos (BDD)	08/06/2016	10/06/2016
Tercer Sprint		
Desarrollo de interfaces	13/06/2016	15/06/2016
Desarrollo de C.R.U.D	16/06/2016	20/06/2016
Cuarto Sprint		
Daily Scrum	Diario durante todo el proyecto	
Sprint Review Meeting	21/06/2016	23/06/2016
Sprint Retrospective	24/06/2016	27/06/2016

4.2.2 Especificación de requisitos de software.

Introducción.

En esta sección se enumera los objetivos y el ámbito de aplicación del documento técnico donde se especifica los requisitos para el desarrollo de la aplicación. “Desarrollo de un sistema web para automatizar el registro de inscripciones en eventos deportivos” con la finalidad de facilitar su entendimiento y características que eventualmente tendrá el software.

Propósito.

Esta aplicación está dirigida a las entidades u organizaciones que desarrollan eventos deportivos, el propósito es realizar inscripciones de equipos y sus respectivos equipos con la finalidad de registrarlos, adicionalmente tener un control de ellos en base a roles de usuario.

Alcance.

El alcance del documento de requerimientos abarca los registros de inscripciones que se realizan en cada evento disponible filtrados por fecha con la intención de recolectar información que será usada para la verificación y la participación de cada equipo inscrito, siendo así punto de partida para desarrollar la aplicación y cumplir los objetivos planteados.

Personal involucrado:

Tabla 16 *Jefe del proyecto (SCRUM).*

Nombre	Xavier Mauricio Rea Peñafiel
Rol	Jefe de Proyecto
Categoría profesional	Ingeniero en sistemas computacionales
Responsabilidades	Coordinar las fases de desarrollo de la aplicación ,levantar y hacer cumplir los requerimientos de la aplicación.
Información de contacto	Ibarra,Tel:0986099536, Mail: mrea@utn.edu.ec

Tabla 17 *Administrador Docker del proyecto (SCRUM).*

Nombre	Milton Andrés Simbaña Alarcón
Rol	Administrador Docker

Categoría profesional	Estudiante de la carrera de ingeniería en sistemas computacionales.
Responsabilidades	Verificar el ambiente de desarrollo para la aplicación y proporcionar información para acceder a la base de datos y al servidor web.
Información de contacto	Ibarra, Tel:0991011494 , Mail: masimbaniaa@utn.edu.ec

Tabla 18 *Administrador de Base de Datos del proyecto (SCRUM).*

Nombre	Milton Andrés Simbaña Alarcón
Rol	Administrador de Base de Datos (BDD)
Categoría profesional	Estudiante de la carrera de ingeniería en sistemas computacionales.
Responsabilidades	Modelar e implementar la BDD para el desarrollo de la aplicación.
Información de contacto	Ibarra, Tel:0991011494 , Mail: masimbaniaa@utn.edu.ec

Tabla 19 *Analista de sistemas del proyecto (SCRUM).*

Nombre	Milton Andrés Simbaña Alarcón
Rol	Analista de Sistemas –Técnico – Programador
Categoría profesional	Estudiante de la carrera de ingeniería en sistemas computacionales.

Responsabilidades	Implementar la aplicación propuesta.
Información de contacto	Ibarra, Tel:0991011494 , Mail: masimbaniaa@utn.edu.ec

Descripción general.

En esta sección se presenta una visión global que describe el desarrollo de la aplicación con el fin de conocer las principales funcionalidades que debe realizar, datos asociados las restricciones impuestas y otros factores que afecten al desarrollo del software a realizarse.

Perspectiva del producto.

Actualmente las entidades u organizaciones que se dedican a crear eventos deportivos o que lo hacen, pero en un periodo de tiempo determinado. No cuentan con un sistema único, que ayude a gestionar el proceso de inscripciones:

- Se realiza petición de información de las inscripciones basados en fechas.
- Se realiza petición de información de los equipos a inscribirse.
- Se realiza petición de información de que jugadores están inscritos y en qué equipo.
- Se recibe información verbalmente.
- Se revisa la información obtenida y se realiza una visión general para el desarrollo de software.
- Se analiza el alcance que podrá tener el software.

En primera instancia de desarrollo el software permitirá a las entidades u organizaciones saber que equipos están inscritos y en que fechas lo hicieron, cuando se creó el evento, definir reglas de inscripción según lo requiera, también podrá definir roles de usuario, adicionalmente saber a qué equipo pertenece cada jugador y en que categoría está inscrito.

Funcionalidades del producto.

En esta sección se describe las funcionalidades más importantes del proyecto que son:

- Diseño de la base de datos para la aplicación.
- Diseño de la aplicación.
- Permitirá gestionar usuarios.
- Permitirá crear eventos
- Permitirá hacer inscripciones de equipos
- Permitirá seleccionar una categoría de inscripción.
- Permitirá asignar jugadores al equipo.

Características de los usuarios:

Tabla 20 *Usuario administrador del sistema (SCRUM).*

Tipo de usuario	Administrador
Habilidades	Administrador total del sistema con ciertos privilegios.
Actividades	Crear eventos, verificar información, añadir usuarios de acuerdo al rol que exista.

Tabla 21 *Usuario organizador del sistema (SCRUM).*

Tipo de usuario	Organizador
Habilidades	Tiene el control de ciertas partes del sistema.

Actividades	Tendrá el privilegio de crear eventos eliminarlos o modificarlos si lo requiere, además de poder asignar un cronograma de juegos usando información de los equipos.

Tabla 22 *Usuario normal del sistema (SCRUM).*

Tipo de usuario	Usuario
Habilidades	Tiene el control de ciertas partes del sistema.
Actividades	Tendrá la posibilidad de crear un equipo en una categoría o varias categorías si lo desea y de forma adicional añadir jugadores según lo requiera.

Restricciones.

Como se ha especificado anteriormente, para el desarrollo de la aplicación se ha usado la metodología SCRUM, las fases de análisis, diseño, arquitectura, modelamiento y despliegue tendrán una etapa inicial para su verificación.

La base de datos y las herramientas usadas para la implementación de la aplicación serán:

- Infraestructura virtual usando la herramienta Docker
- Se desplegarán de Docker 2 contenedores uno de ellos contendrá el servidor de aplicaciones Tomcat y el otro tendrá la base de datos PostgresSQL dentro de sí.

- Para el desarrollo de la aplicación JSF se requieren tres librerías, en este caso la librería Primefaces v5.3, Bootsfaces v0.8.6 y finalmente la librería de conexión con la base de datos hacia la aplicación que es postgresql-9.2-1002.jdbc4

Evolución previsible del sistema.

Inicialmente el sistema podrá realizar inscripciones de equipos, agregar jugadores en diferentes categorías establecidas y en la parte administrativa crear roles de usuario, crear eventos asignando fechas, asignar un cronograma de nivel uno con los equipos inscritos.

Eventualmente el proyecto de software propuesto será adaptable a dispositivos móviles y adicionalmente cargada a Google Play que permitirá ofrecer esta aplicación para el SO de Android definiéndola así un portal ideal para integrar esta aplicación con diversos dispositivos móviles (De Luca, 2014).

En cuanto a funcionalidad contendrá más roles de usuarios como el usuario arbitraje el cual podrá asignar resultados por medio de un Smartphone, también logrará generar cronogramas de nivel dos en adelante de acuerdo a los resultados obtenido por el arbitraje, adicionalmente podrá añadir información acerca de los jugadores como faltas, el mejor goleador, el mejor encestador etc.

Requisitos funcionales.

- **Adecuación:** el proyecto de software debe proporcionar una diversidad de funciones para varias tareas que se pueden ejecutar dentro del sistema, en esté sistema como se define en el alcance de plan de proyecto engloba solo ciertas funciones prioritarias del sistema.
- **Exactitud:** el proyecto de software debe proporcionar los resultados correctos o acordados.
- **Interoperabilidad:** el proyecto de software tendrá la capacidad para interactuar con uno o más sistemas especificados.

Aplicación.

Tabla 23 *Aplicación número de requisito RF.APL.01.*

Número de requisito	RF.APL.01
Nombre del requisito	Gestión de usuarios
Tipo	☒Requisito ☐Restricción
Detalle del requisito	El usuario podrá gestionar usuarios según lo requiera.
Prioridad del requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Tabla 24 *Aplicación número de requisito RF.APL.02.*

Número de requisito	RF.APL.02
Nombre del requisito	Límite de acceso a usuarios
Tipo	☒Requisito ☐Restricción
Detalle del requisito	En el desarrollo de este proyecto existirán 3 tipos de usuarios: **Administrador. -** es el encargado de crear y administrar los roles de usuario de la misma forma también gestionara la parte de eventos deportivos. **Organizador. –** este usuario se encargará de crear eventos eliminarlos o modificarlos según lo requiera, podrá visualizar todos los equipos, además se encargará de generar el cronograma de nivel uno para definir enfrentamientos de equipos la hora y el lugar donde se debe realizar el evento.

	Usuario.- tienen acceso al sistema exclusivamente para realizar sus inscripciones en diferentes eventos creados previamente por el usuario organizador, además podrá crear un equipo y añadir jugadores al mismo, todo esto en las categorías que se haya definido.
Prioridad del requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Tabla 25 *Aplicación número de requisito RF.APL.03.*

Número de requisito	RF.APP.03
Nombre del requisito	Crear un reporte de inscripciones dependiendo del evento creado.
Tipo	☒Requisito ☐Restricción
Detalle del requisito	Todos los usuarios inscritos y sus jugadores serán parte de un reporte generado según el evento en el que se inscribieron.
Prioridad del requisito	☒Alta/Esencial ☐Media/Deseado ☒Baja/Opcional

Datos de entrada.

Tabla 26 *Datos de entrada número de requisito RF.ENT.01.*

Número de requisito	RF.ENT.01
Nombre del requisito	Ingreso de la información
Tipo	☒Requisito ☐Restricción
Detalle del requisito	Se hará la petición de información básica en los formularios de la aplicación con la intención de manejar esa información con diferentes usuarios.
Prioridad del requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Requisitos no funcionales.

Arquitectura

Los requisitos no funcionales se refieren a los requisitos necesarios para implementar la aplicación JSF.

Tabla 27 *Arquitectura número de requisito RNF.ARQ.01.*

Número de requisito	RNF.ARQ.01
Nombre del requisito	Requisitos de hardware y software
Tipo	☒Requisito ☐Restricción
Detalle del requisito	Para el desarrollo de esta aplicación se construyó una infraestructura virtual en un servidor usando la herramienta Docker bajo Linux, en el cual se instaló dos contenedores ,en

	uno de ellos la Base de Datos PostgresSQL y en el otro el servidor de aplicaciones Tomcat , la ejecución de esta aplicación será en cualquier navegador web en su versión más reciente.
Prioridad del requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Tabla 28 *Arquitectura número de requisito RNF.ARQ.02.*

Número de requisito	RNF.ARQ.02
Nombre del requisito	Carga de datos
Tipo	☒Requisito ☐Restricción
Detalle del requisito	La carga de datos es la parte más importante del sistema ya que de ello dependerá el sistema y sus funcionalidades, guardara y mostrara información de acuerdo a cada usuario a cada momento.
Prioridad del requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Usabilidad.

La usabilidad ayuda al usuario dando especificaciones del sistema con la intención de que sea entendido, aprendido, operado, usable y adaptable a cualquier dispositivo. (De Luca, 2014)

Tabla 29 *Usabilidad número de requisito RNF.USA.01.*

Número de requisito	RNF.USA.01
Nombre del requisito	Acceso a la aplicación JSF
Tipo	☒Requisito ☐Restricción
Detalle del requisito	La aplicación será desplegada usando un servidor virtual de modo que se podrá acceder inicialmente por una IP de servidor más el puerto por el que se ha levantado el servicio de Tomcat según se haya especificado.
Prioridad del requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Seguridad.

Especificaciones destinadas a la capacidad del sistema para proteger la información de manera que las personas o sistemas no autorizados no puedan leerlos o modificarlos.

Tabla 30 *Seguridad número de requisito RNF.SEG.01.*

Número de requisito	RNF.SEG.01
Nombre del requisito	Seguridad de ingreso a la aplicación
Tipo	☒Requisito ☐Restricción
Detalle del requisito	Los usuarios internos y externos para acceder a la aplicación deberán utilizar un usuario y contraseña.
Prioridad del requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Tabla 31 *Seguridad número de requisito RNF.SEG.02.*

Número de requisito	RNF.SEG.02
Nombre del requisito	Seguridad en base de datos
Tipo	☒Requisito ☐Restricción
Detalle del requisito	La seguridad de la base de datos estará bajo la responsabilidad del administrador de base de datos y adicionalmente del administrador de Docker.
Prioridad del requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

Tabla 32 *Seguridad número de requisito RNF.SEG.03.*

Número de requisito	RNF.SEG.03
Nombre del requisito	Contenedores Docker
Tipo	☒Requisito ☐Restricción
Detalle del requisito	La herramienta Docker brinda seguridad al 100% ya que para levantar un contenedor se deben especificar muchos aspectos en el cual se ejecutará una aplicación , también al ser un servidor propio la seguridad de datos es mayor, todos estos aspectos definirá el administrador Docker.
Prioridad del requisito	☒Alta/Esencial ☐Media/Deseado ☐Baja/Opcional

4.2.3 Arquitectura de software.

Propósito.

El propósito de realizar esta sección de arquitectura de software es presentar a los participantes del proceso de desarrollo de la aplicación una visión clara y concisa sobre los aspectos del proyecto de software en base a la arquitectura del sistema con el objetivo de facilitar la comprensión del mismo.

Visión Global.

En esta sección se detalla la arquitectura del sistema donde se mostrará distintas vistas como los casos de uso, el modelamiento de la base de datos entre otros aspectos relevantes que forman parte del desarrollo del sistema web.

Metas y Restricciones Arquitectónicas.

Con el desarrollo del sistema de automatización de inscripciones para eventos deportivos se busca obtener una aplicación con un alto nivel de portabilidad, inicialmente la aplicación será alojada bajo los contenedores de la herramienta Docker con la intención de comprobar que la infraestructura virtual que ofrece esta herramienta es apta, segura y eficaz para el alojamiento de aplicaciones, es por eso se alojara la aplicación JSF en Docker, y adicionalmente obtener de Bitnami el servidor web Tomcat y la base de datos Postgres el cual ayudarán al desarrollo de esta aplicación usando el patrón MVC, se tiene como menta lograr que la infraestructura virtual construida no solo sea para este tipo de aplicaciones en este lenguaje sino para todos los lenguajes de programación posibles que se encuentren dentro de las limitaciones Bitnami.

Vista de Casos de uso.

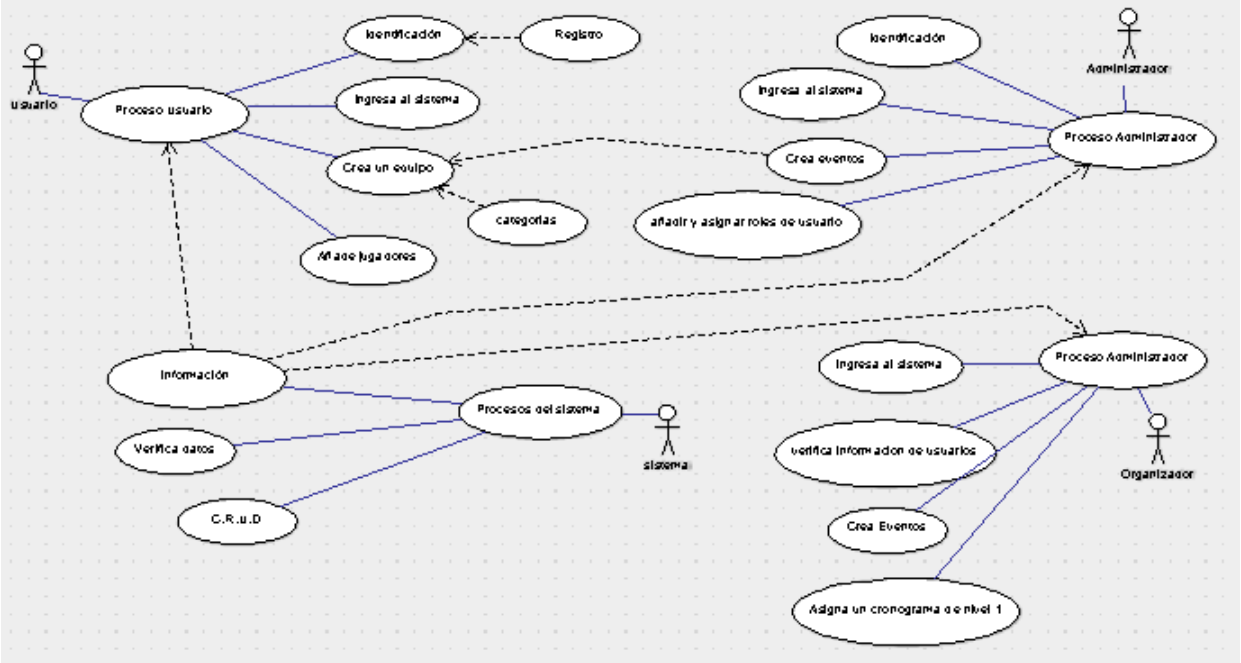


Figura 91. Caso de Uso del Sistema.

Vista de Datos.

Diagrama entidad relación (ER)

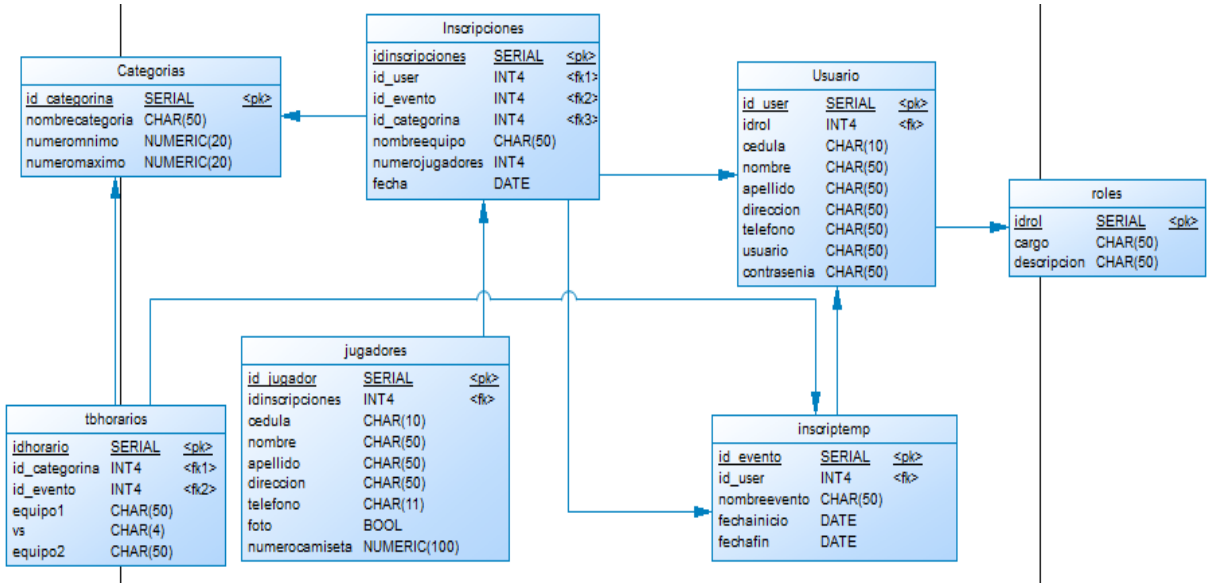


Figura 92. Modelo entidad relación del sistema.

Vista de implementación.

Arquitectura de software.

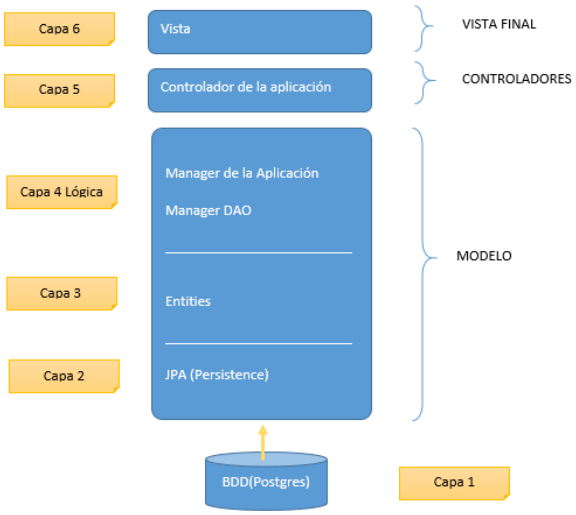


Figura 93. Arquitectura JSF.

Arquitectura Docker.

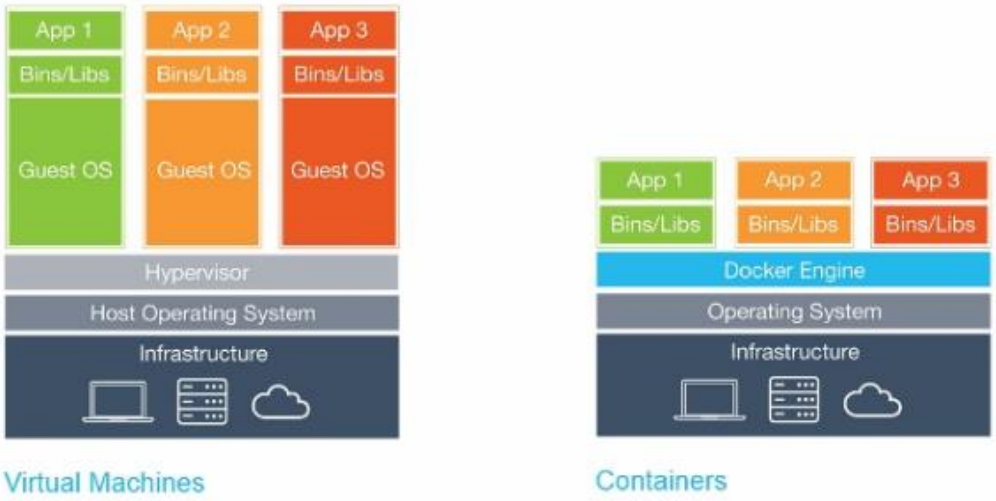


Figura 94. Arquitectura Docker .*(encamina, 2016)*

4.2.4 Implementación de la aplicación.

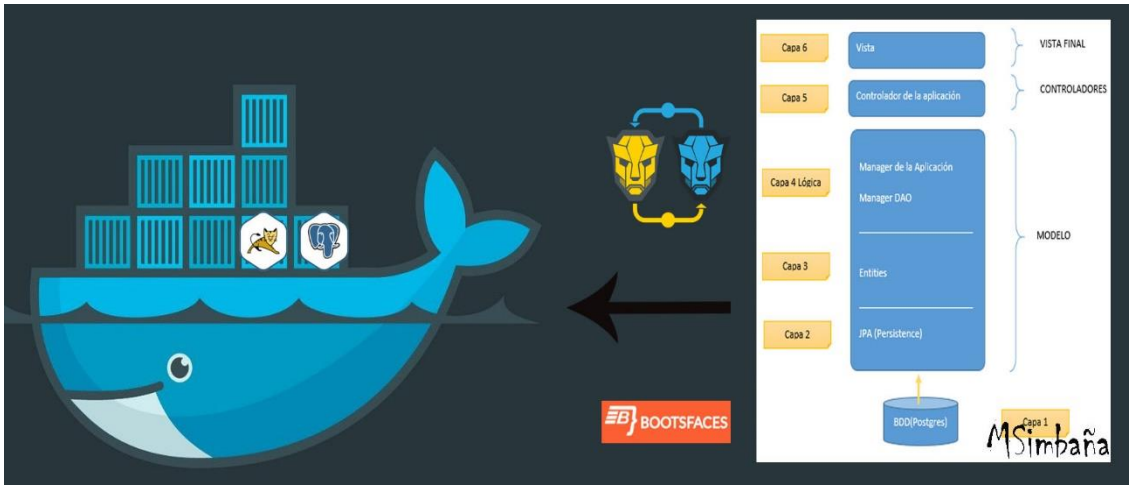


Figura 95. Visión global de implementación de la aplicación.

Crear una infraestructura virtual.

Como punto de inicio se realiza la instalación de Docker en un sistema operativo Linux en este caso se realizó dicha instalación en Ubuntu 14.04, también se obtuvo las imágenes que brinda Bitnami compatibles con Docker, usando específicamente las imágenes que contienen instalaciones de Tomcat y PostgresSQL, con estas herramientas se construirá un ambiente para alojar la aplicación a desarrollarse.

Figura 96. Pull de imágenes Bitnami.

Tabla 33 *Tamaño de la herramienta Docker y componentes.*

Nombre de la imagen Virtual Docker	Tamaño del archivo	Versión
Tomcat	426.8 MB	8.0.36
Postgres	259.8 MB	9.6
Docker	2.849 MB	1.11.0

Aplicación JSF desplegada en Docker.

Java Server faces (JSF) es un framework para aplicaciones Java con el cual se realizó el sistema web de inscripciones en eventos deportivos, ciertamente para el diseño de esta aplicación se usó la tecnología XHTML el cual fusiona las tecnologías HTML y XML para obtener como resultado a XHTML, también se usó librerías de diseño con Bootsfaces y libreas de conexión a la base de datos. (Torres Remón, 2014)

Entonces para iniciar a realizar el despliegue de la aplicación de debe seguir los siguientes pasos:

- Al tener previamente creada la infraestructura virtual en Docker ejecutar las imágenes de la BDD y del servidor web Tomcat.
- Usar el comando imágenes de docker para mostrar que imágenes contiene docker.

```
docker@docker-VirtualBox:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
ubuntu              14.04               b72889fa879c        11 weeks ago
 188 MB
ubuntu              latest              b72889fa879c        11 weeks ago
 188 MB
bitnami/postgresql  latest              a2c378db4670        12 weeks ago
 403.6 MB
bitnami/tomcat      latest              6465d9dbc945        3 months ago
 610.3 MB
docker@docker-VirtualBox:~$
```

Figura 97. Verificación de imágenes Docker para la aplicación.

- Con el siguiente comando ejecutar la BDD de postgres especificando el puerto, un nombre de contenedor y por último el usuario, contraseña y la base de datos que vamos a usar.

```
docker@docker-VirtualBox:~$ docker run -d -p 9090:5432 --name dbtesisv4 -e POSTG
RES_USER=dbmsimba -e POSTGRES_PASSWORD=1003862792 -e POSTGRES_DB=juegos_db bitna
mi/postgresql
```

Figura 98. Ejecución de la BDD PostgresSQL.

En este caso se define que para la conexión del servidor de BDD postgres desde un cliente con las siguientes credenciales.

Tabla 34 *Credenciales para la conexión de PostgresSQL.*

Credenciales para cliente postgres	
Puerto	9090
Usuario	dbmsimba
Contraseña	1003862792
Base de datos	Juegos_db
IP del servidor	192.168.56.3

- Verificar que la imagen docker se encuentra en ejecución usando el siguiente comando: docker ps.

```
docker@docker-VirtualBox:~$ docker run -d -p 9090:5432 --name dbtesisv5 -e POSTGRE
S_USER=dbmsimba -e POSTGRES_PASSWORD=dockercliente -e POSTGRES_DB=juegos_db bitnam
i/postgresql
c7baa7c962ed870131683d795aa52646cea980780a601e56c75f308ec354fbc0
docker@docker-VirtualBox:~$ docker ps
CONTAINER ID        IMAGE                 COMMAND             CREATED             S
TATUS               PORTS                    NAMES
c7baa7c962ed        bitnami/postgresql    "/entrypoint.sh"    8 minutes ago       U
p 8 minutes         0.0.0.0:9090->5432/tcp   dbtesisv5
docker@docker-VirtualBox:~$
```

Figura 99. Contenedores en ejecución y listos para usar.

- Al finalizar se tiene todo listo para ingresar a la DBB desde un cliente postgres, para continuar este proceso tener previamente instalado postgres en el cliente del servidor Docker, en este caso el cliente tiene el sistema operativo Windows 10 donde se encuentra instalado PostgresSQL la versión 9.4, entonces de dicha instalación se realizó la conexión al servido de la Base de datos Postgres situado en el servidor Docker.

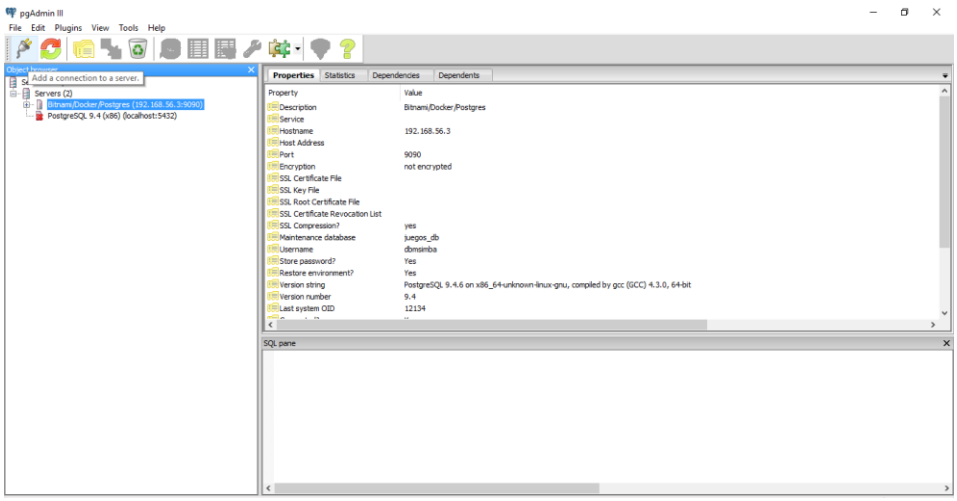


Figura 100. Pgadmin III en Windows 10.

- Una vez dentro del Pgadmin III de postgres se realizó la conexión usando la opción de conexión de postgres en la parte superior izquierda.

Figura 101. Icono de conexión Pgadmin III.

- Al hacer clic en este icono se despliega la siguiente ventana que pide las credenciales para acceder a un servidor de Base de datos; llenar los campos con la información requerida.

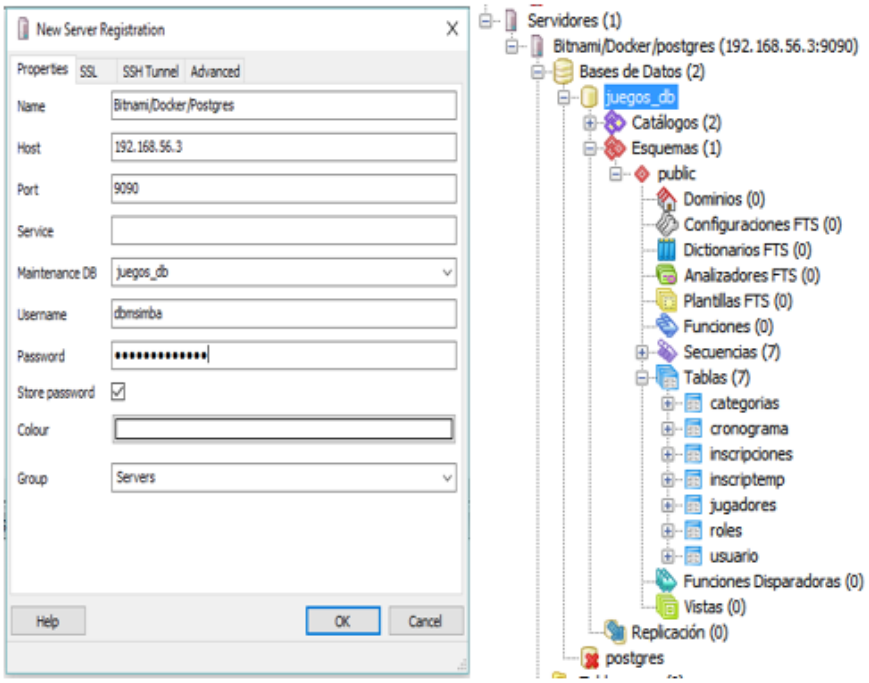


Figura 102. Inserción de credenciales y verificación de la conexión a la BDD.

- Al finalizar se tiene la conexión realizada de tal manera que se la use normalmente para la aplicación desarrollada previamente, para alojar la aplicación hacer ciertas configuraciones en el proyecto, el primer paso es obtener el archivo .war del proyecto que se lo obtiene realizando un clean and build en el proyecto.

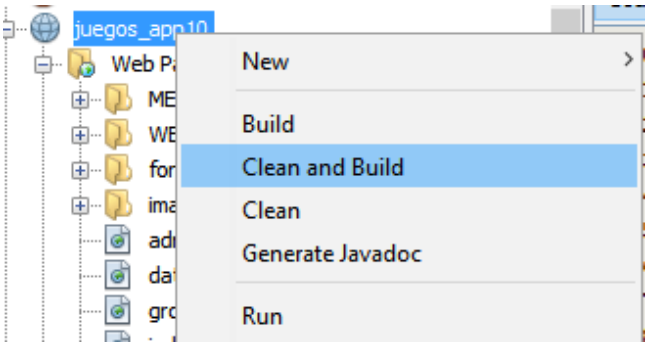


Figura 103. Clean a build del proyecto realizado.

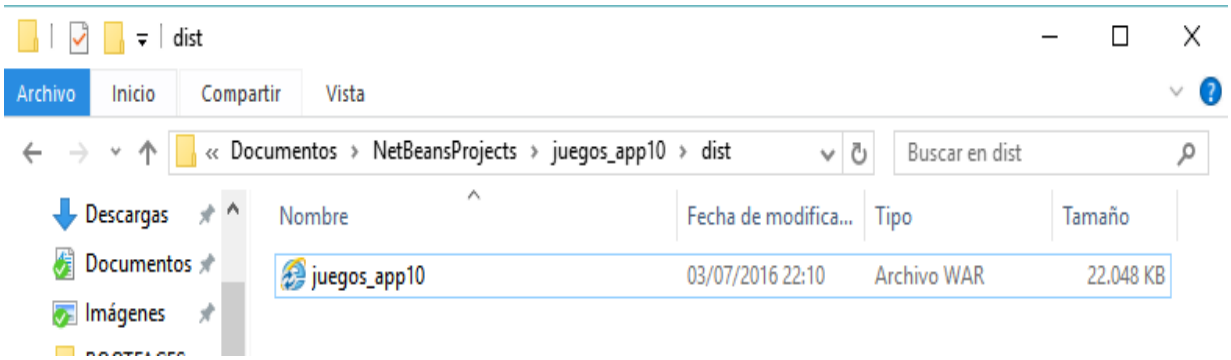


Figura 104. Archivo de extensión **.war** del proyecto.

- Al obtener este archivo de extensión **.war** del proyecto, copiar dicho archivo en el servidor de docker o en el lugar donde tengamos acceso al servidor Tomcat.

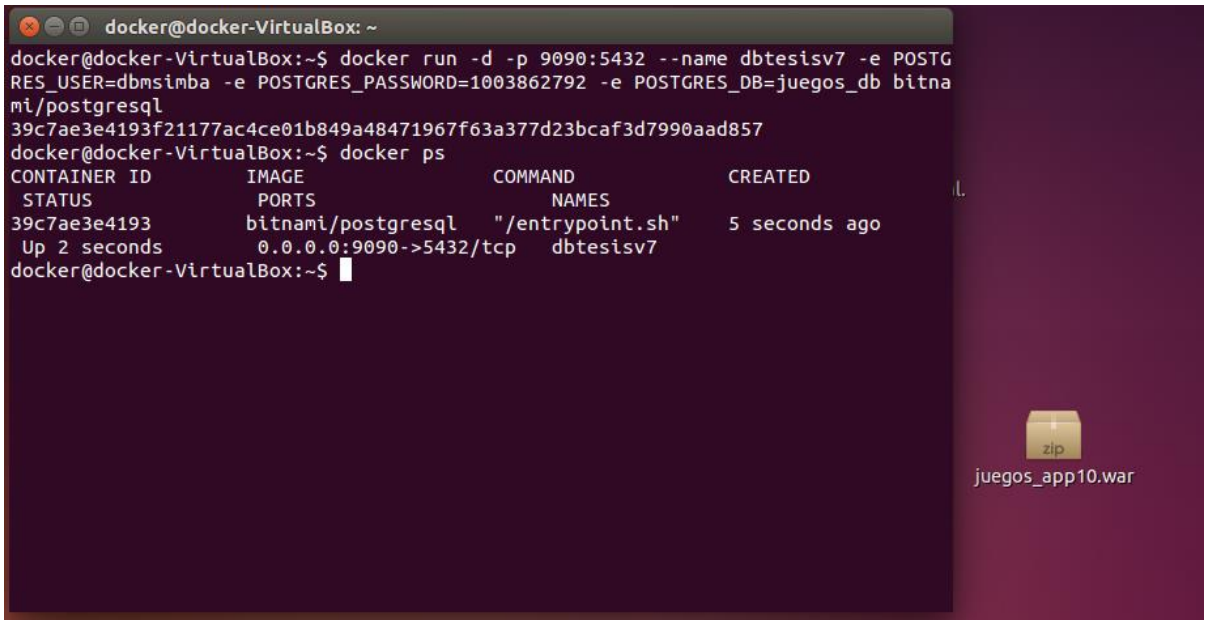


Figura 105. Copia del archivo .war al servidor Ubuntu Docker.

- Doble clic en el archivo .war para verificar si el percistence.xml de conexión con la BDD está configurada correctamente caso contrario editar de acuerdo a las credenciales de conexión con postgres.

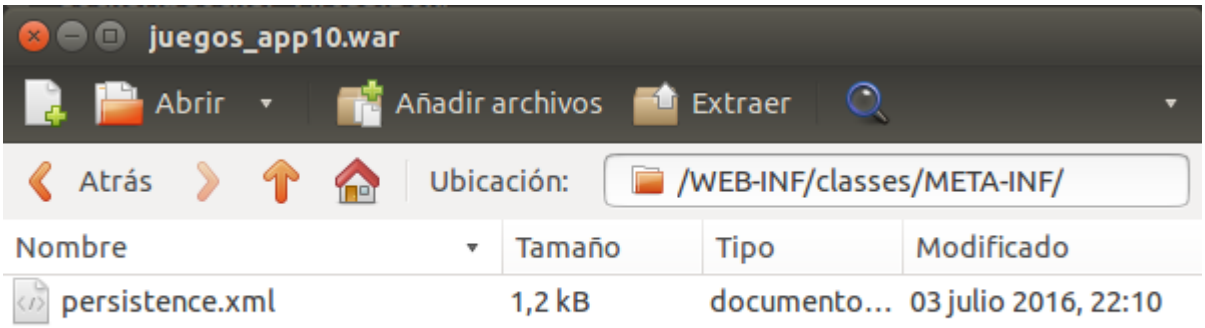


Figura 106. Ubicación del archivo persistence.xml dentro del proyecto.

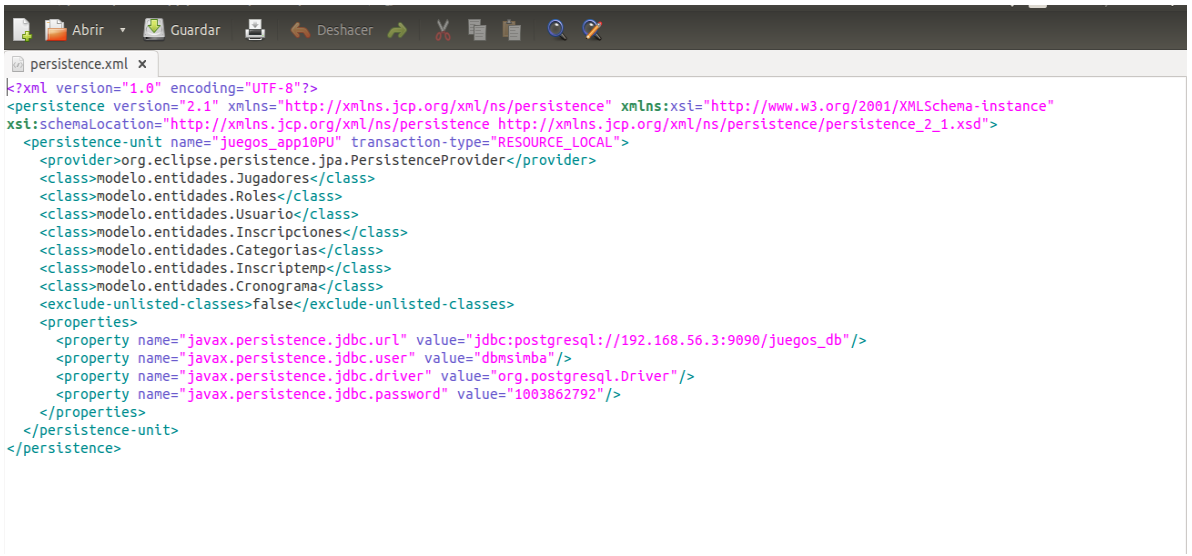


Figura 107. Contenido del archivo persistencia.xml.

- Una vez que todo está correcto proceder a desplegar la aplicación en el servidor Tomcat de Docker, es decir se debe ejecutar la imagen de Tomcat con el siguiente comando:

```
docker@docker-VirtualBox:~$ docker run -d -P --name web12 --link dbtesisv7 -e TO
MCAT_USER=juegos -e TOMCAT_PASSWORD=juegosapp bitnami/tomcat
d360b779975bbc0e8b1ff09a998218055f9fa481303a270a15386295390f7f6f
docker@docker-VirtualBox:~$
```

Figura 108. Ejecución del servidor Tomcat.

En este comando se especifica el puerto, nombre de usuario, contraseña y adicionalmente el enlace hacia la base de datos.

- Verificar que se ejecutó el servidor Tomcat en Docker con el comando docker ps.

```
docker@docker-VirtualBox:~$ docker ps
CONTAINER ID        IMAGE                COMMAND                  CREATED
 STATUS              PORTS                    NAMES
d360b779975b        bitnami/tomcat       "/entrypoint.sh"         3 minutes ago
 Up 3 minutes        0.0.0.0:32768->8080/tcp   web12
39c7ae3e4193        bitnami/postgresql   "/entrypoint.sh"         About an hour ago
 Up About an hour    0.0.0.0:9090->5432/tcp    dbtesisv7
docker@docker-VirtualBox:~$
```

Figura 109. Contenedores en ejecución para la aplicación.

Tomcat tendrá el puerto 32768 y la base de datos el puerto 9090 por el cual se establece una conexión para la gestión de los mismos.

- Ejecutar Tomcat en el navegador para verificar que está funcionando y subir el proyecto con extensión .war.

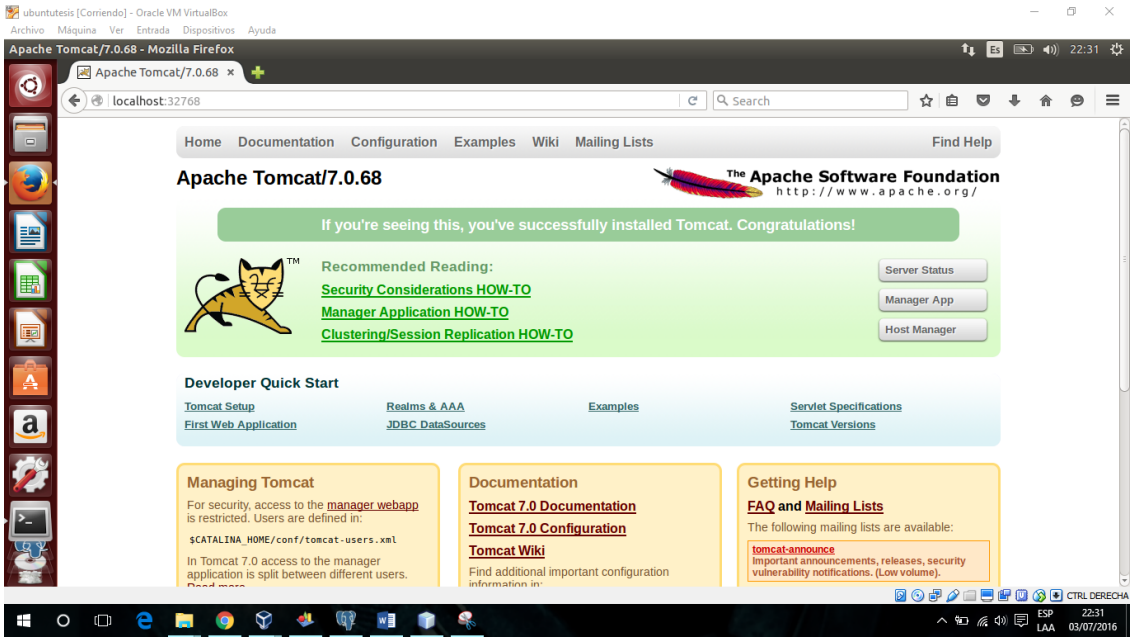


Figura 110. Vista del contenido del contenedor Tomcat.

- Ingresar las credenciales en la opción manager app con las respectivas credenciales de usuario y contraseña que se fue especificada al crear el contenedor de Tomcat.

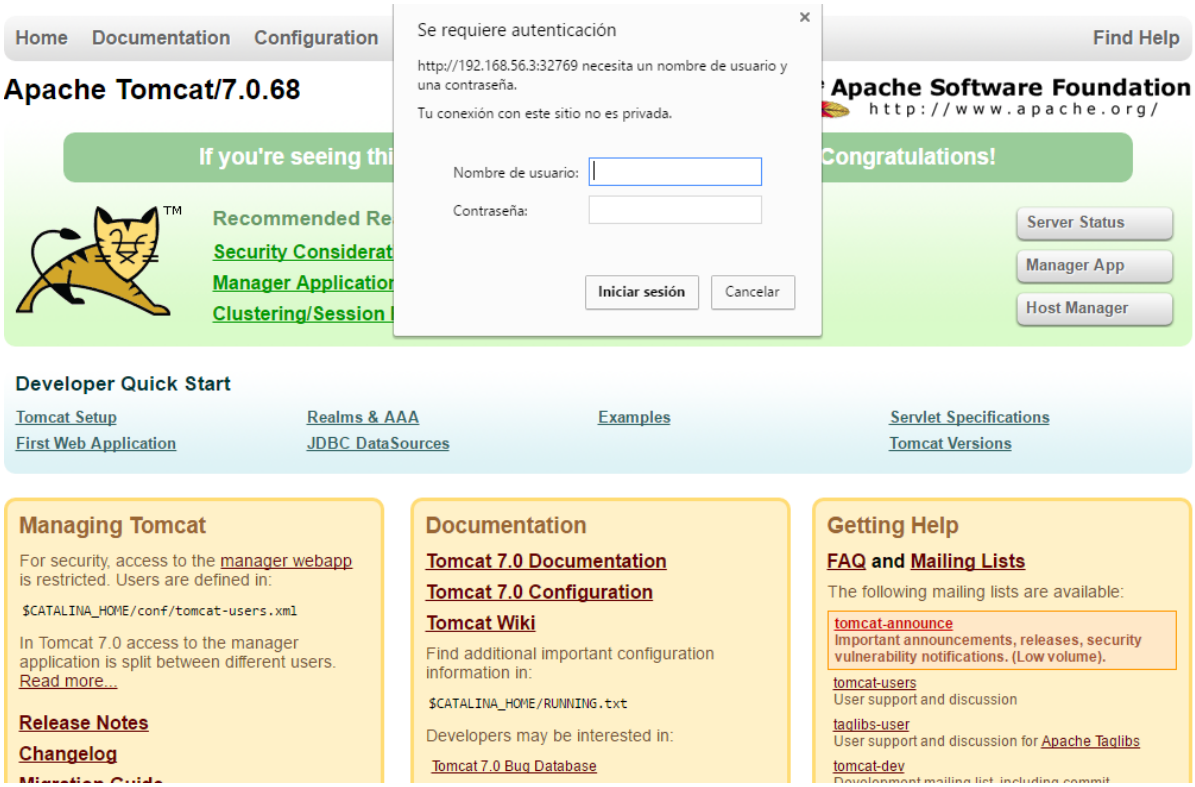


Figura 111. Autenticación en Tomcat.

- Al ingresar a Tomcat buscar la opción de Deploy; allí se puede subir la aplicación de extensión .war que contiene el proyecto que se ha realizado.

Figura 112. Búsqueda de la sección para subir el archivo .war del proyecto.

- De esta forma el software realizará la petición de donde se encuentra el archivo .war para proceder a cargarlo en Tomcat.

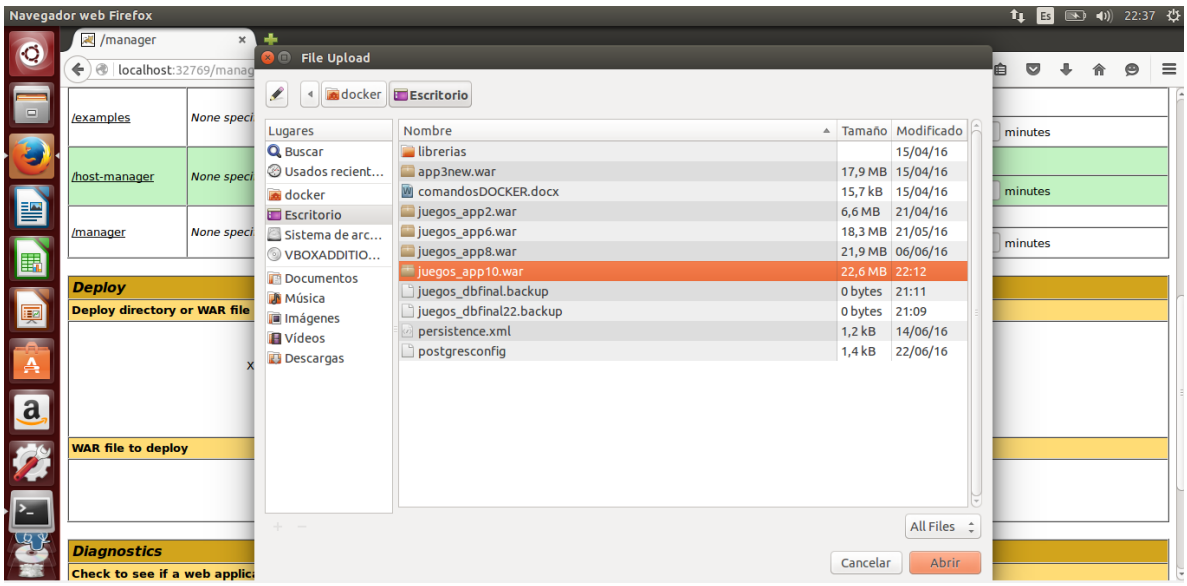


Figura 113. Selección archivo .war para cargar en Tomcat.

- Una vez que se ha cargado el proyecto en Tomcat hacer clic en el botón Deploy.

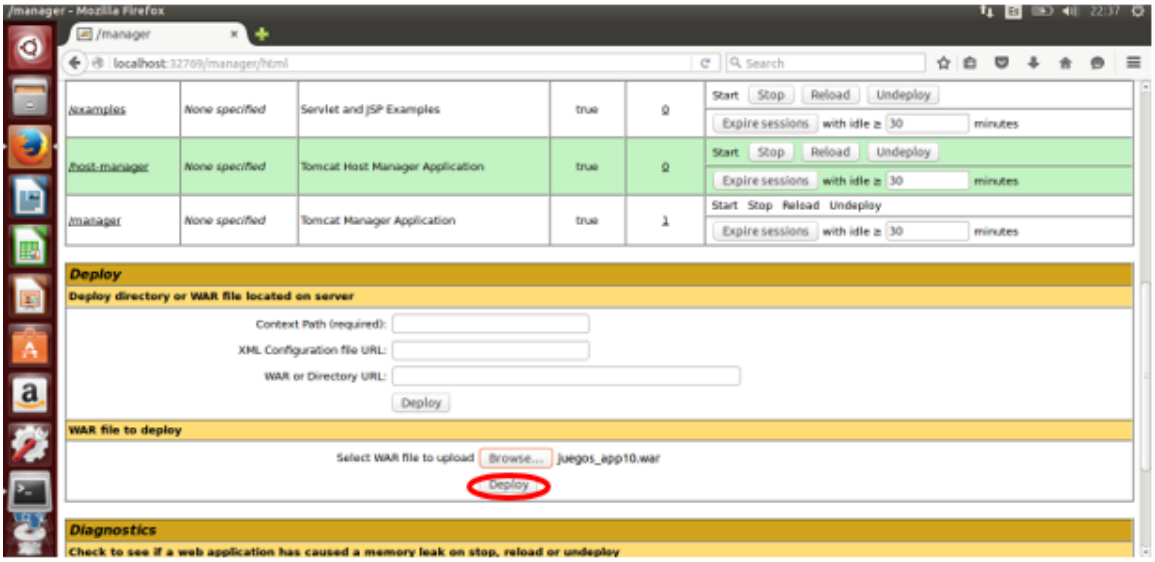


Figura 114. Realizar un despliegue del proyecto **.war.**

/juegos_app10	*Ninguno especificado*		true	2	Arrancar Parar Recargar Replegar Expirar sesiones sin trabajar ≥ 30 minutos

Figura 115. Verificación de que el proyecto se encuentra en Tomcat.

- Al finalizar todo este procedimiento hacer clic en la aplicación subida anteriormente o en nuestro navegador preferido ingresamos la dirección IP de servidor, el puerto de Tomcat y finalmente el nombre de la aplicación.

192.168.56.3:32769/juegos_app10/

Figura 116. Dirección donde se encuentra la aplicación en ejecución.

Finamente se tiene una aplicación de prueba alojada en la infraestructura virtual creada previamente como se muestra a continuación:

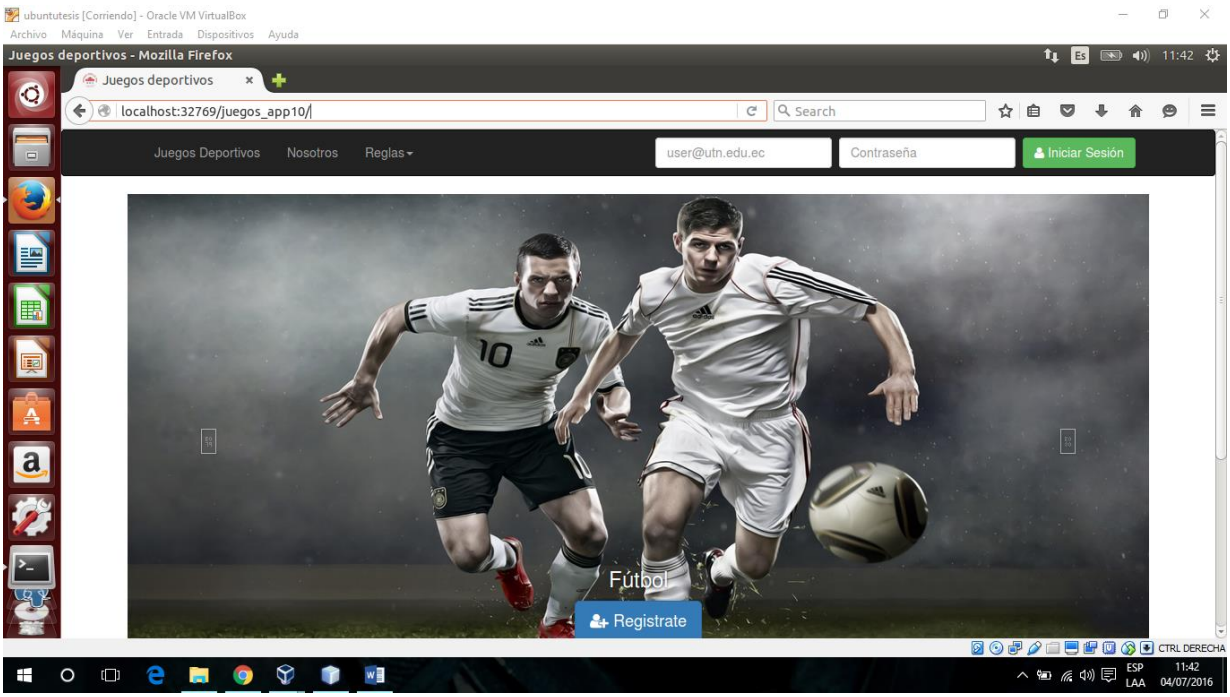


Figura 117. Ejecución del proyecto desde el servidor Docker.

En esta aplicación realiza la petición de un usuario y contraseña si no lo tiene simplemente el usuario podrá registrarse y asumir el rol de un usuario normal.

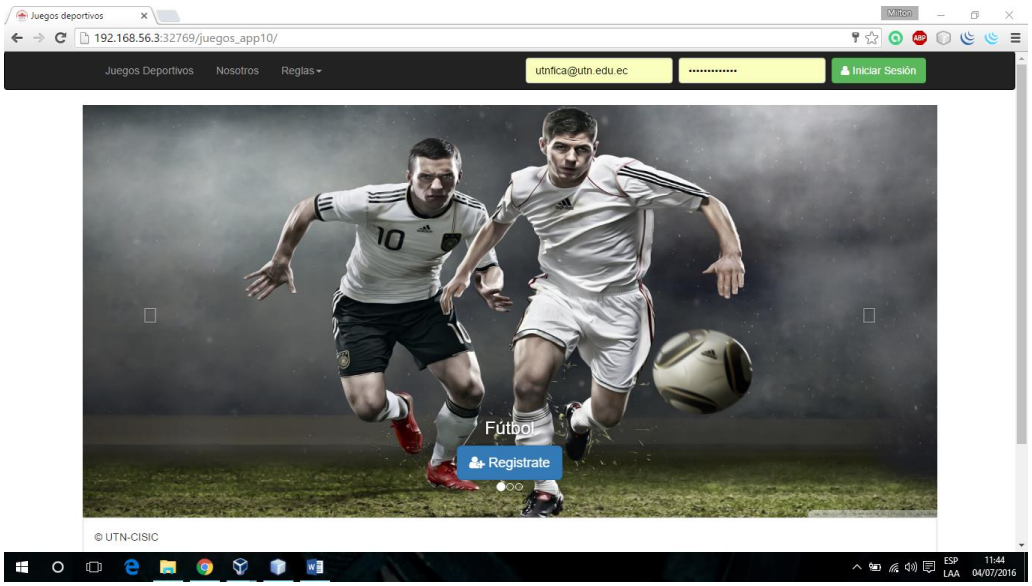


Figura 118. Ejecución de la aplicación desde el cliente Windows.

Inicia con la vista del usuario administrador el cual puede crear un evento y gestionar los usuarios de la aplicación según lo requiera.

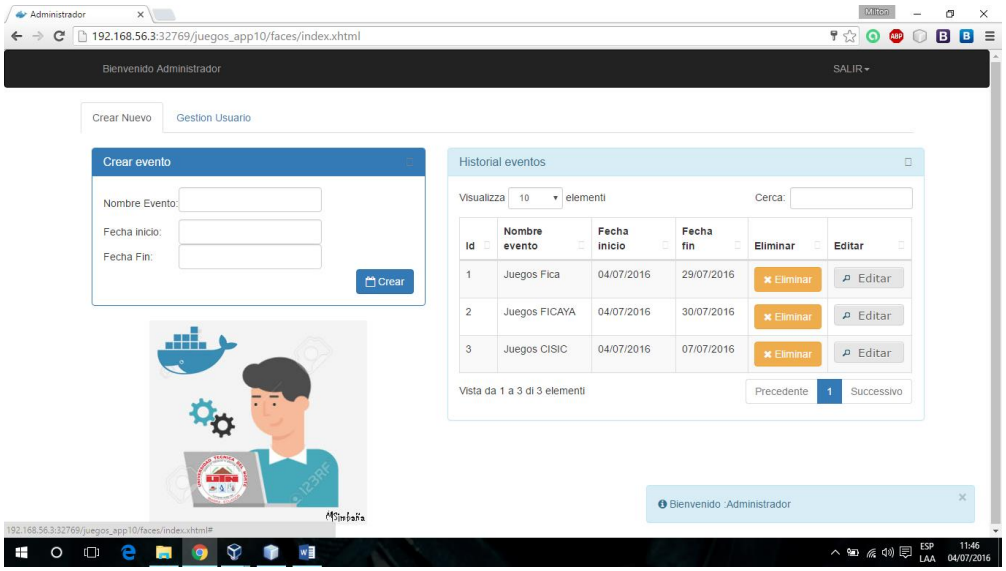


Figura 119. Vista rol administrador del proyecto 1.

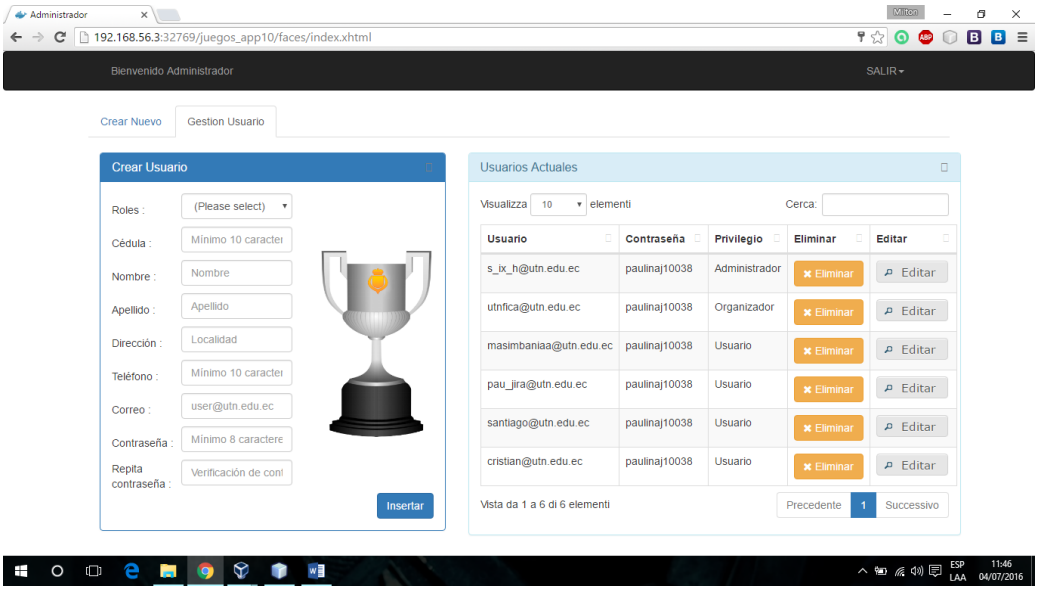


Figura 120. Vista rol administrador del proyecto 2.

Seguido del usuario administrador tendremos el usuario Organizador el cual además de crear un evento podrá generar un cronograma de nivel uno para los enfrentamientos según el evento y la categoría.

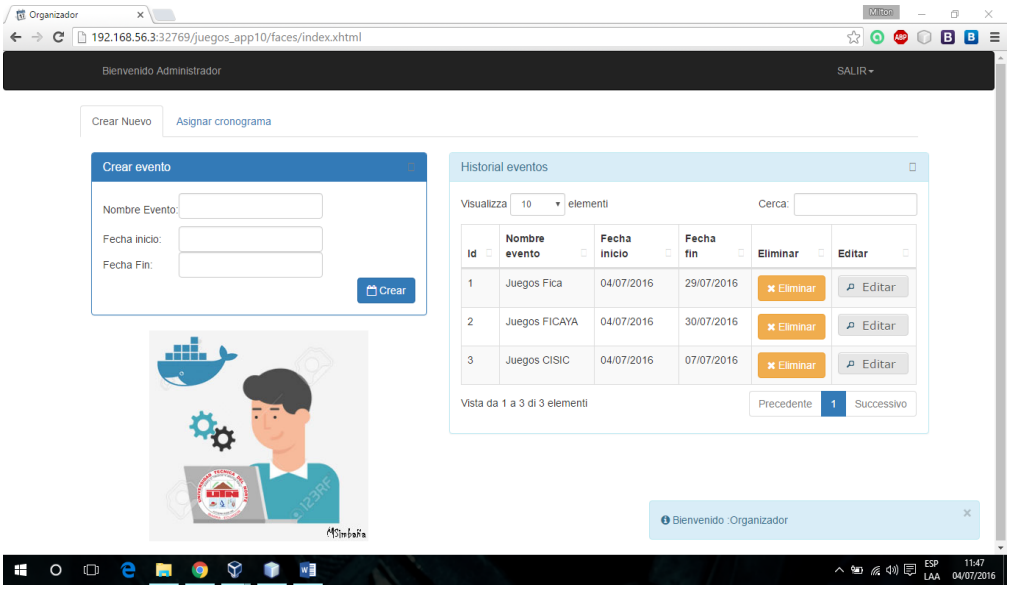


Figura 121. Vista rol organizador del proyecto 1.

El usuario Administrador es capaz de realizar un filtro por evento y categoría definiendo así los equipos a enfrentarse y generar el cronograma de nivel 1.

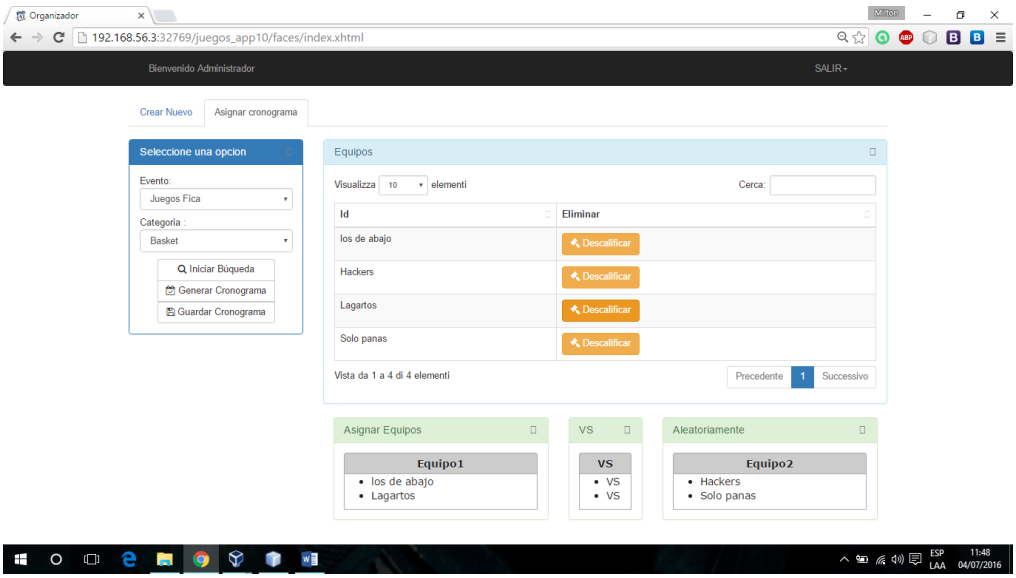


Figura 122. Vista rol organizador del proyecto 2.

Finalmente tendremos el usuario normal, este podrá visualizar que evento está disponible para inscribir el equipo y los jugadores.

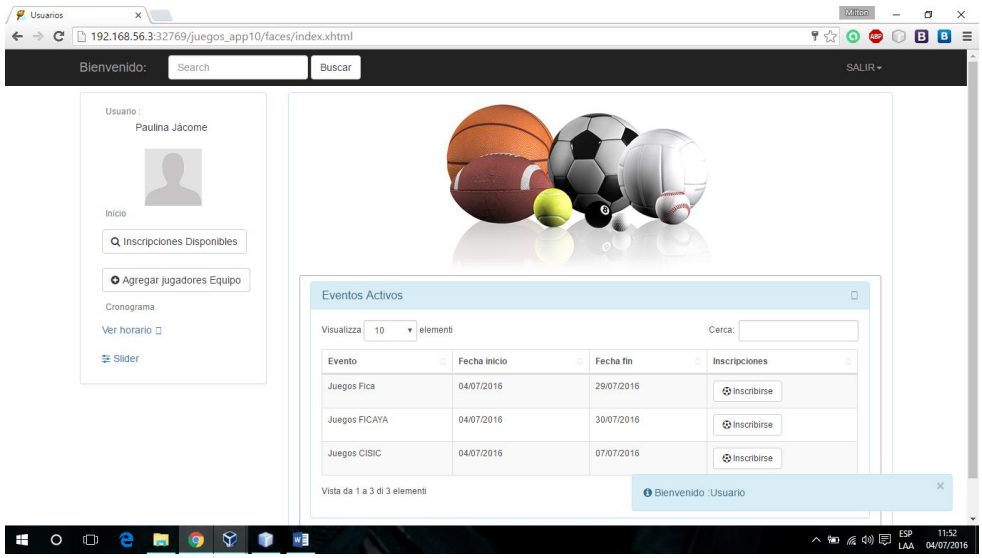


Figura 123. Vista rol Usuario 1.

El usuario puede inscribir su equipo estableciendo como parámetros el nombre, número de jugadores y categoría.

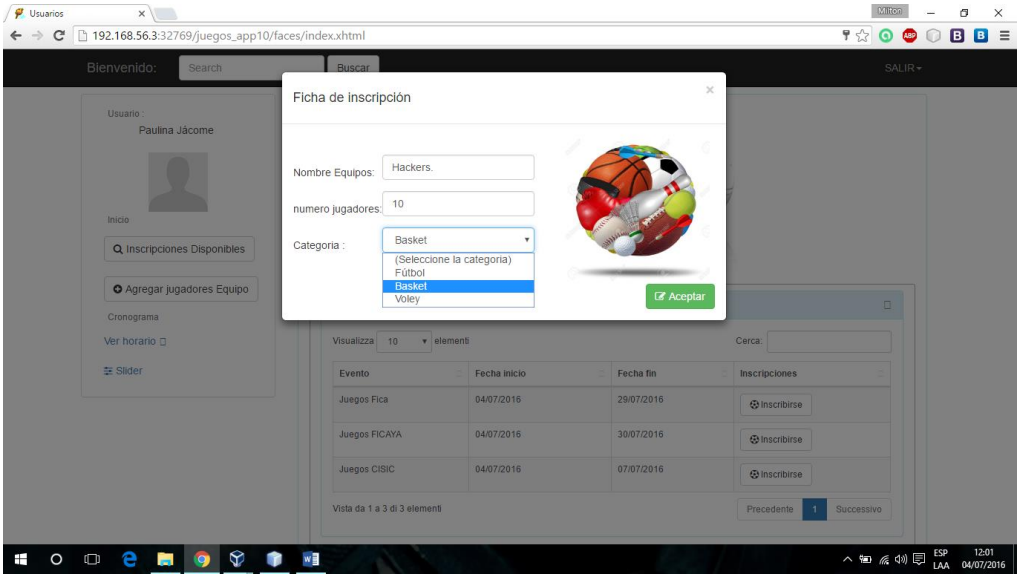


Figura 124. Vista rol Usuario 2.

También muestra el equipo que está inscrito y agrega jugadores según la necesidad, además podrá ver los datos de los jugadores dependiendo del evento y categoría inscritos.

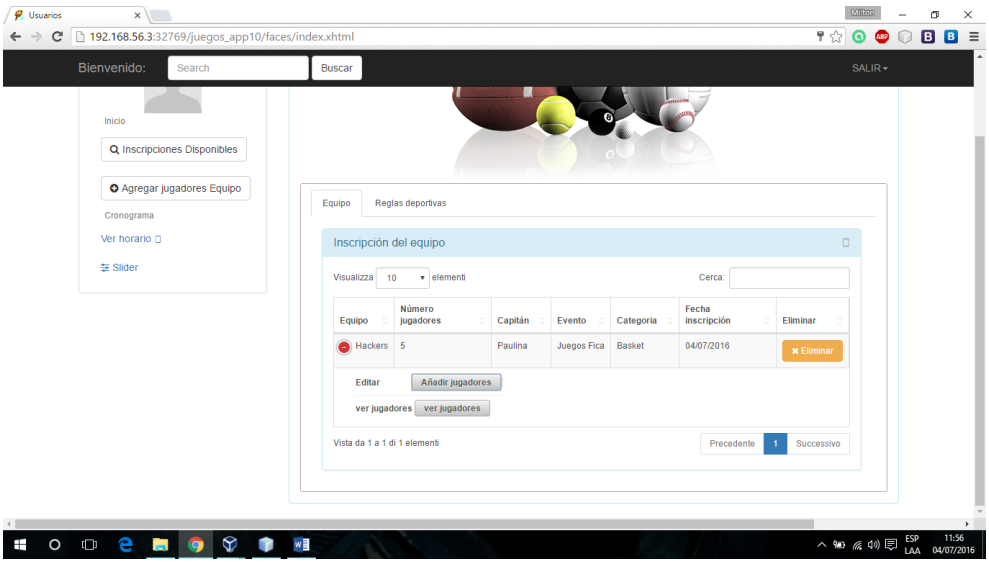


Figura 125. Vista rol Usuario 3.

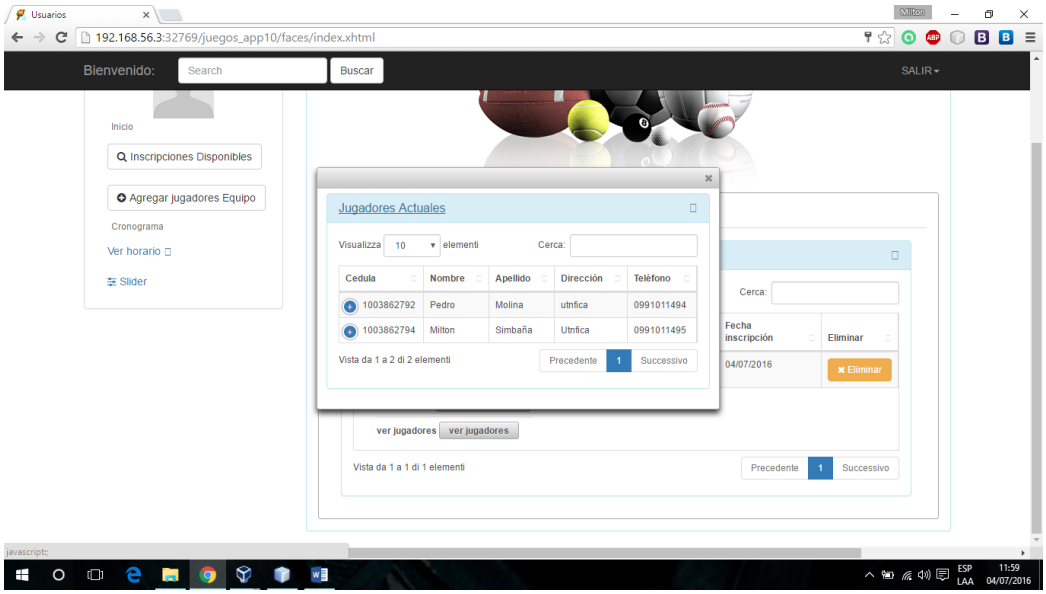


Figura 126. Vista rol Usuario 4.

En esta sección de la aplicación también se realizó un botón que contenga la regla de los juegos con la finalidad de que los usuarios tengan conocimiento de ellas.

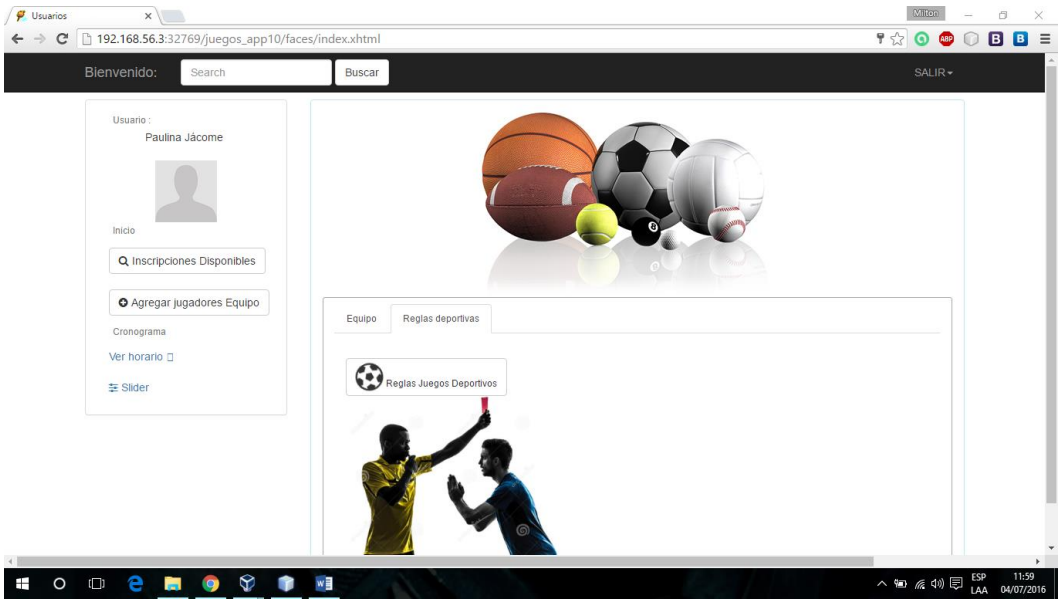


Figura 127. Vista rol Usuario 5.

Propuesta de implementación en la plataforma FICA.

Este estudio se realizó en base al alojamiento de aplicaciones en una plataforma local con el fin de aligerar, optimizar y administrar el funcionamiento de las misma, actualmente en los servidores de la Facultad de ingeniería en ciencias aplicadas (FICA), cuentan con servidores que contienen el sistema operativo Linux la versión de Centos 6.5 en el cual se alojan varias aplicaciones en un mismo lenguaje de programación y de la misma manera no existe una gestión administrativa de las aplicaciones, el objetivo de este estudio es poder implementar esta nueva tecnología en los servidores FICA con la intención de mejorar la agilidad, concurrencia, administración y ligereza de aplicaciones y servicios web, también se realizó la instalación de la herramienta docker en un sistema operativo Linux Centos 7 x64 para demostrar que la herramienta es multiplataforma y que se puede usar dentro de la los servidores FICA manteniendo así el estándar a nivel de servidor que se maneja en la facultad.

Instalación de Docker en Centos 7

La instalación de docker es rápida, esta se debe realizar bajo súper usuario y se usan las siguientes instrucciones para su instalación.

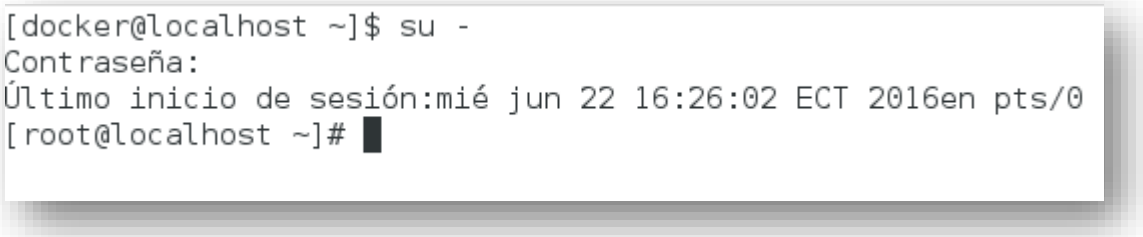


Figura 128. Súper usuario en Centos 7

```
sudo yum update
curl -fsSL https://get.docker.com/ | sh
sudo service docker start
sudo docker run hello-world
```

Figura 129. Comandos para la de instalación Docker en Centos 7

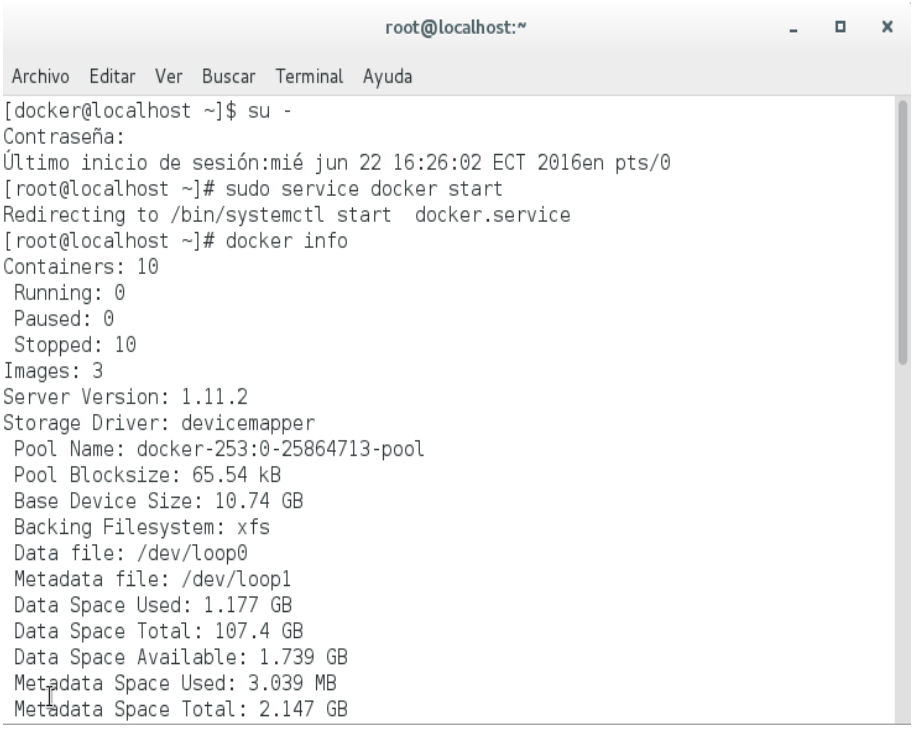


Figura 130. Inicialización de Docker en Centos 7.

Al finalizar la instalación en Centos se usan las mismas instrucciones que se establecieron en Ubuntu con la única diferencia que en Ubuntu no se necesita ser súper usuario para llevar acabo la instalación, además cada vez que el equipo se apague o se reinicie en Centos se deberá reiniciar el servicio de docker para usarlo con normalidad y no tener ningún problema.

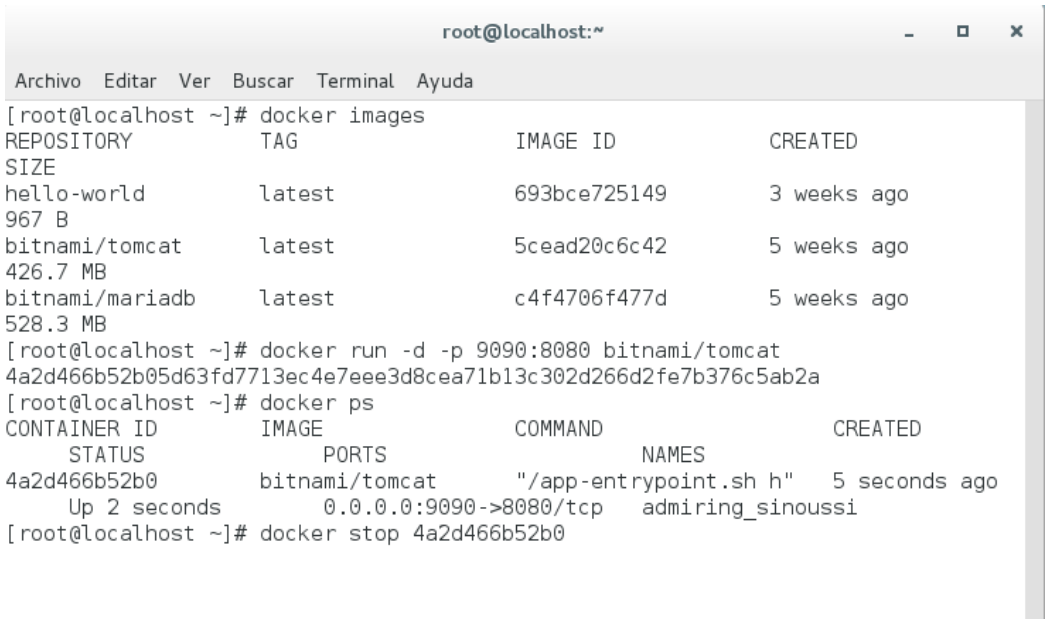


Figura 131. Ejecución y verificación de la imagen Tomcat en Centos 7.

Finalmente se ejecuta un navegador para verificar que el servicio es este caso de Tomcat está habilitado.

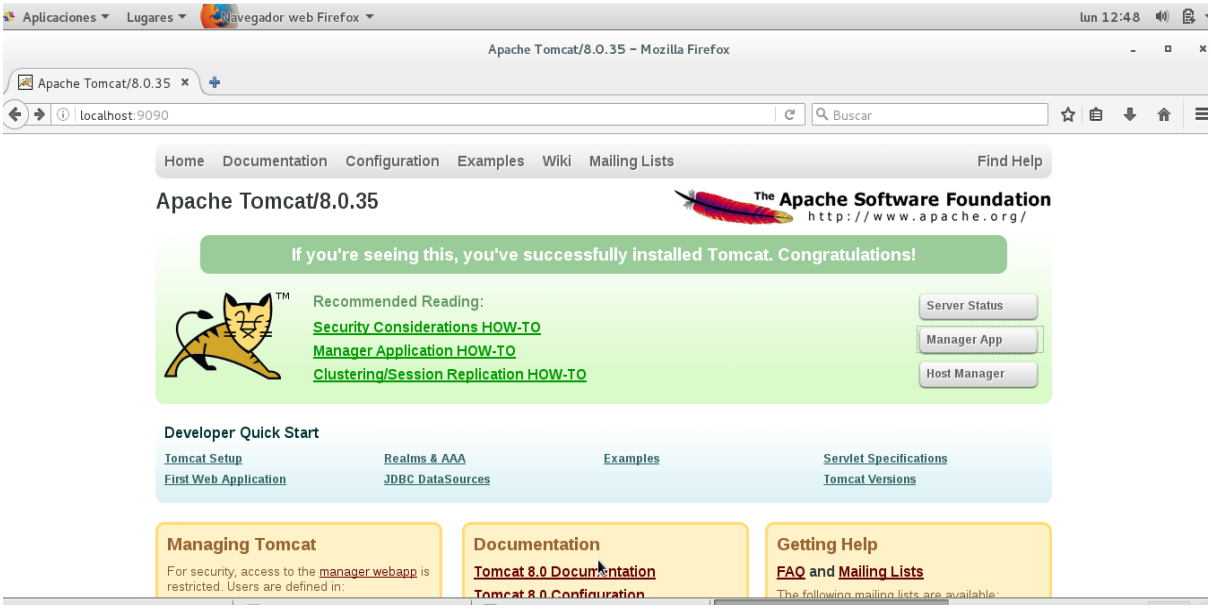


Figura 132. Contenedor Tomcat en Centos 7.

4.3 Análisis de resultados

Esta herramienta es súper ligera y sumamente fácil de instalar ya que solo bastan tres instrucciones para lograrlo, también dentro de sí podremos tener imágenes docker que contienen instalaciones independientes con los que podremos generar los contenedores Docker.

Bitnami es una herramienta que proporcionó las imágenes docker de una forma fácil y segura, además esta herramienta brinda soporte total a todas las imágenes docker que se obtuvieron a través de GitHub, es por eso que la herramienta Bitnami en una pieza fundamental para el desarrollo de infraestructuras virtuales.

Se considera que no solo se realizó la instalación de la herramienta Docker sino también se ha verificado su funcionalidad con el objetivo de proponer el uso de esta herramienta en la plataforma FICA, adicionalmente tras desarrollar la aplicación de prueba para alojarla en una infraestructura virtual, se determinó que esta herramienta es compatible con varios Frameworks y bases de datos con lo cual se posibilita la construcción de una data center usando esta tecnología.

El desarrollo de la aplicación permitió demostrar que esta tecnología funciona correctamente y que puede ser usada en grandes entidades ayudando a gestionar diversas aplicaciones mejorando su rendimiento.

Para confirmar la agilidad y concurrencia de Docker se usó la herramienta Apache-JMeter la cual ayudo a realizar pruebas de rendimiento y se determinó que la herramienta Docker es óptima y efectiva de la siguiente manera.

- Una vez ejecutada la herramienta JMeter procedemos a realizar una serie de pasos para poder crear un nuevo proyecto el cual probará la concurrencia con 1000 usuarios.

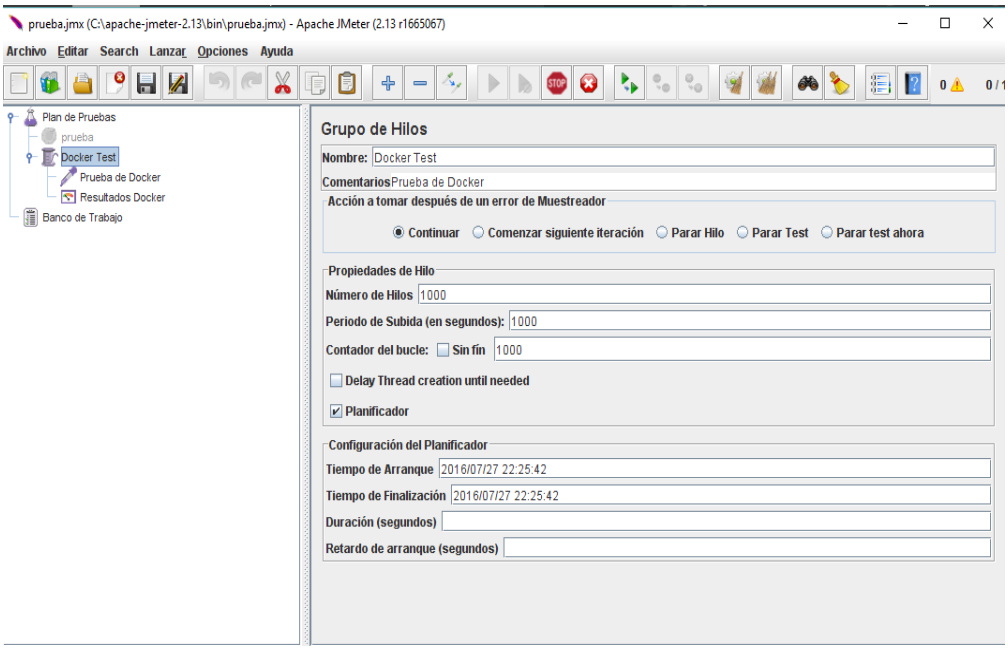


Figura 133. Pruebas de Docker-JMeter 1.

- Se asigna la dirección IP que tiene la página web y el puerto por que brinda el servicio.

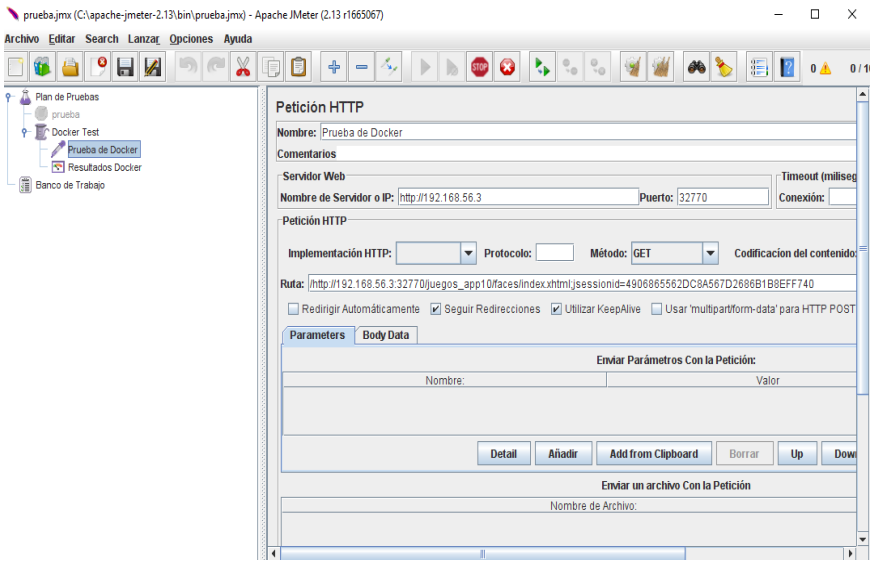


Figura 134. Pruebas de Docker-JMeter 2.

- Los resultados se muestran de forma inmediata en una tabla para proceder posteriormente a analizarlos.

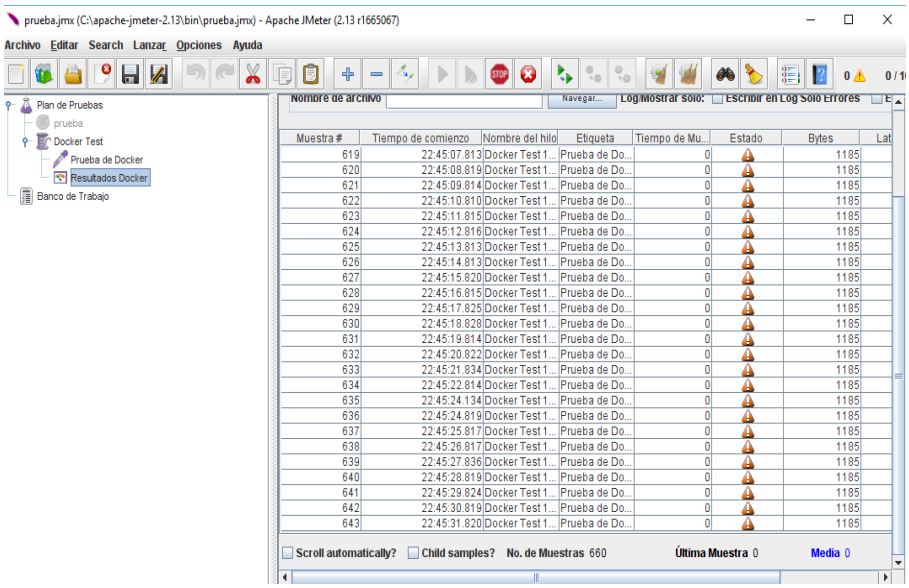


Figura 135. Pruebas de Docker-JMeter 3.

- Según la herramienta usada se comparan los siguientes tiempos de ejecución de docker junto a las herramientas de instalación tradicionales.

Tabla 35 *Comparación de ejecución de herramientas.*

Herramienta	Tiempo de ejecución Docker	Tiempo de ejecución Tradicional
Tomcat + Aplicación	1minuto :40 segundos	4 minutos :30 segundos
PostgresSQL + BDD	1minuto:10 segundos	2 minuto:50 segundos

Los contenedores en ejecución analizados con la herramienta JMeter ayudaron a evidenciar que el rendimiento es óptimo, de forma que las peticiones desde un cliente al servidor son inmediatas a comparación de una petición tradicional, además de mostrar al administrador de docker que tamaño, nombre e identificación del contenedor se está ejecutando.

4.4 Resumen de resultados

Una vez realizada la instalación de la herramienta Docker, se realizó el uso de la herramienta Bitnami como complemento para desarrollar la infraestructura virtual que se necesitaba para el alojamiento de la aplicación, Bitnami es una herramienta que facilito y remplazo a los Dockerfiles que son claves para el desarrollo de una infraestructura virtualizada ya que con ellos se puede crear imágenes docker, de este modo Bitnami crea dockerfiles personalizados de una forma estándar para el uso de Docker, es por eso que Bitnami proporciona a Docker varias imágenes compatibles y que pueden ser obtenidas a través de GitHub.

Bitnami proporciona las imágenes PostgresSQL y Tomcat los cuales forman parte de la infraestructura virtual que alojaran una aplicación.

Para estas imágenes se crearon ciertas credenciales para acceder desde un cliente a cada uno de los contenedores con el propósito de poder gestionarlos según lo necesitemos.

Al finalizar de construir la infraestructura virtual en docker y alojar la aplicación en los contenedores, Docker presenta ciertas características administrativas entre contenedores e imágenes docker ayudando así a gestionar los contenedores y aplicaciones que se encuentren alojadas.

Los resultados obtenidos por tiempos de ejecución de docker frente a las herramientas tradicionales propone adoptar esta tecnología en los servidores de la facultad FICA con el objetivo de aislar diferentes infraestructuras virtuales rompiendo así barreras y limitaciones de lenguajes de programación.

CAPITULO 5

Conclusiones y recomendaciones

5.1 CONCLUCIONES

5.2 RECOMENDACIONES

5.3 REFERENCIAS Y BIBLIOGRAFÍA

5.3.1 REFERENCIAS WEB.
5.3.2 BIBLIOGRAFÍA.

5.4 ANEXOS

5.1 Conclusiones

- Docker es una tendencia tecnológica que podrían adoptar muchas entidades con el fin de mejorar el rendimiento en sus aplicaciones y disminuir constes en hardware y software.
- Para este estudio la implementación de docker no presentó limitaciones en su software libre en cuanto a su funcionalidad, de este modo permitió cumplir totalmente con los objetivos planteados.
- Debido a las características que presenta docker, se logró crear una infraestructura virtual que posteriormente permitió el alojamiento de aplicaciones web.
- Usar esta tecnología es sencilla y permite publicar aplicaciones de forma rápida debido a que docker usa máquinas virtuales ligeras en Linux, además usa librerías y archivos de configuración solamente si son estrictamente necesarios.
- La aplicación de prueba se la pudo realizar sin mayor problema ya que efectivamente Bitnami fue clave para el desarrollo de la infraestructura virtual que requería la aplicación de prueba desarrollada en JSF.
- Esta herramienta es sumamente ligera para instalar y su funcionamiento es óptimo ya que permite realizar instancias de las imágenes docker para crear contenedores de este modo podremos alojar varias aplicaciones con su propia infraestructura virtual.
- El crecimiento de esta herramienta es extraordinario ya que a finales de este año 2016 presento el desarrollo de imágenes que contienen herramientas CRM el cual eventualmente se podrá implementar estudios relacionados con los clientes de las empresas.

5.2 Recomendaciones

- Es recomendable seguir con el estudio de esta herramienta y las imágenes que brinda Bitnami con opción a implementación a un servidor dedicado privado, con el objetivo de intentar construir un centro de datos privado usando la herramienta docker y Bitnami.
- Bitnami ayudó a demostrar que Docker funciona y que se puede crear infraestructuras para cualquier lenguaje de programación que este dentro de las limitaciones de Bitnami, ya que esta herramienta continúa dando soporte a docker con sus instalaciones independientes, es por ello que recomiendo usar la herramienta docker junto a Bitnami.
- Docker y Bitnami son herramientas tecnológicas que buscan la optimización agilidad y sobre todo mejorar la concurrencia en las aplicaciones de esta forma se recomienda implementar estas tecnologías en la plataforma FICA.
- Se recomienda realizar aplicaciones de prueba en los diversos lenguajes de programación que estén dentro de las limitaciones de Bitnami con el fin de alojarlas en Docker demostrando el alcance de la herramienta.
- Al instalar la herramienta en la versión de Linux Centos 7 se deberá usar al usuario su, también se deberán habilitar los puertos por el cual ejecutemos una imagen de docker en el caso de que no se encuentren habilitados.

5.3 Referencias y bibliografías

5.3.1 Referencias web.

Docker. (2016). Docker. [online] Available at: https://www.docker.com/

Bitnami.com. (2016). Bitnami Container Images for Docker. [online] Available at: https://bitnami.com/docker/

OpenWebinars.net. (2014). Docker, Qué es y sus principales características. [online] Available at: https://openwebinars.net/docker-que-es-sus-principales-caracteristicas

Picodotdev.github.io. (2015). *Aplicaciones multicontenedor con Docker Compose | Blog Bitix*. [online] Available at: https://picodotdev.github.io/blog-bitix/2015/07/aplicaciones-multicontenedor-con-docker-compose

Joseangelfernandez.es. (2015). Preguntas frecuentes sobre Docker para usuarios de Windows – joseangelfernandez.es. [online] Available at: http://www.joseangelfernandez.es/blog/2015/03/preguntas-frecuentes-sobre-docker-para-usuarios-de-windows

Blog.celingest.com. (2015). Docker y la era de los contenedores | Celingest Blog – Feel the Cloud. [online] Available at: http://blog.celingest.com/2015/02/17/docker-y-la-era-de-los-contenedores

5.3.2 Bibliografía.

Aguilar, L. J. (2012). *Computación en la nube:ESTRATEGIAS DE CLOUD COMPUTING EN LAS EMPRESAS.* Mexico: Alfaomega.

De Luca, D. (2014). *Apps HTML5 para móviles: Desarrollo de aplicaciones para smartphones y tablets basado en tecnologías web.* México: Alfaomega.

encamina, E. (05 de 07 de 2016). *Introducción a docker*. Obtenido de Por una nube sostenible: http://blogs.encamina.com/por-una-nube-sostenible/2016/04/14/introduccion-a-docker/

Gómez Jiménez, E. (2012). *Desarrollo de software con NetBeans 7.1: Programe para escritorio, Web y dispositivos móviles.* México: Alfaomega.

Hurwitz, J., Bloor, R., & Kaufman, M. (2010). *Cloud Computing For Dummies.* Hoboken,New Jersey: Wiley.

Labs, V. S. (15 de 06 de 2016). *Clickherelabs.com*. Obtenido de Clickherelabs.com: http://clickherelabs.com/2015/05/virtual-scrum-boards/

Livingston, G. (2014). *Cloudcomputingcafe.com*. Obtenido de Finding security and reliability in cloud computing. | Cloud Computing Cafe.

Marinescu, D. C. (2013). *Cloud Computing: Theory and Practice.* USA: EL SEVIER.

Martínez, D. R., Aranda, P. J., & López, Ó. P. (2010). *Aplicaciones Web* . México: Alfaomega.

Peña, R. (2013). *Uso de las TIC en la vida diaria.* México: Alfaomega.

Sosinsky, B. (211). *Cloud Computing Bible.* Indianapolis,India: Wiley.

Torres Remón, M. Á. (2014). *Diseño web con HTML5 y CSS3.* Lima: Macro.

José, J., Condori, M., & Stallman, R. (2011). Ventajas y Desventajas de Cloud Computing. *Revista de Informacion Tecnologica Y Sociedad*, 86–87. Retrieved from http://www.revistasbolivianas.org.bo/pdf/rits/n7/n7a35.pdf

Profesional, A., Diego, I., Arévalo, J., Francisco, D., & Martín, R. (2011). Computación en la nube, 1–25.

Servidores, V. De, Futuro, U. S. De, Doña, J. M., García, J. E., López, J., Pascual, F., & Pascual, R. F. (n.d.). Virtualización de Servidores. Una Solución de Futuro.

5.4 ANEXOS

- **Instalación de Ubuntu 14.04 x64.**

Realizamos la instalación en español y seleccionamos la opción Instalar Ubuntu.

Figura 136. Instalación Ubuntu paso 1.

Verifica características para la instalación de Ubuntu en nuestro equipo y seleccionamos la opción continuar.

Figura 137. Instalación Ubuntu paso 2.

En esta ventana indica en que partición del disco vamos a instalar Ubuntu, si tenemos una partición definida seleccionamos la opción (Más opciones), en mi caso seleccionare (Borrar disco e instalar Ubuntu) y seleccioné la opción instalar ahora.

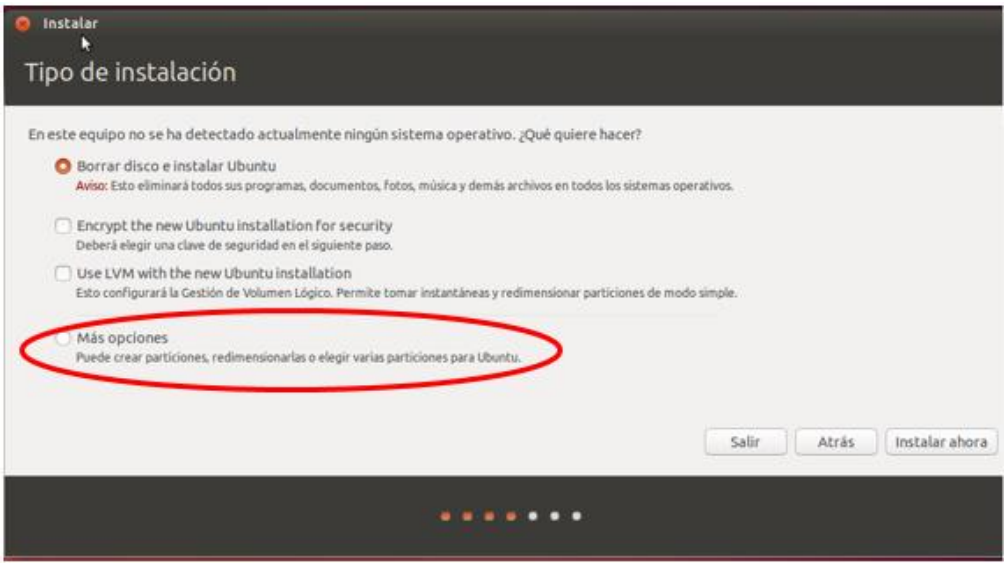


Figura 138. Instalación Ubuntu paso 3.

Seleccionamos continuar para seguir con la instalación.

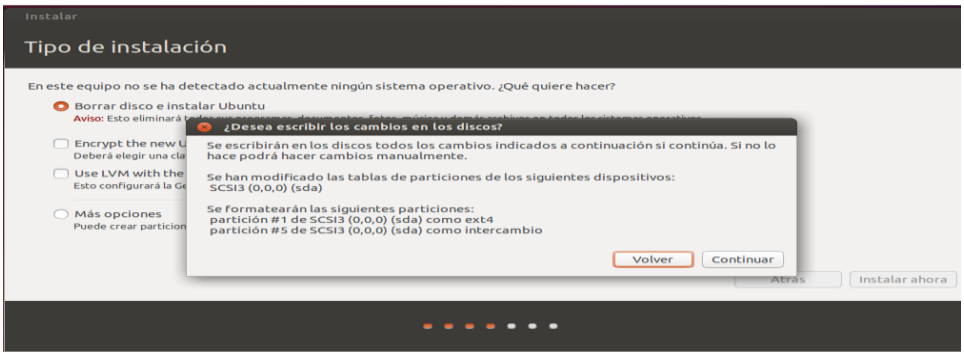


Figura 139. Instalación Ubuntu paso 4.

Seleccionamos la zona donde nos encontremos y seleccionamos la opción continuar.

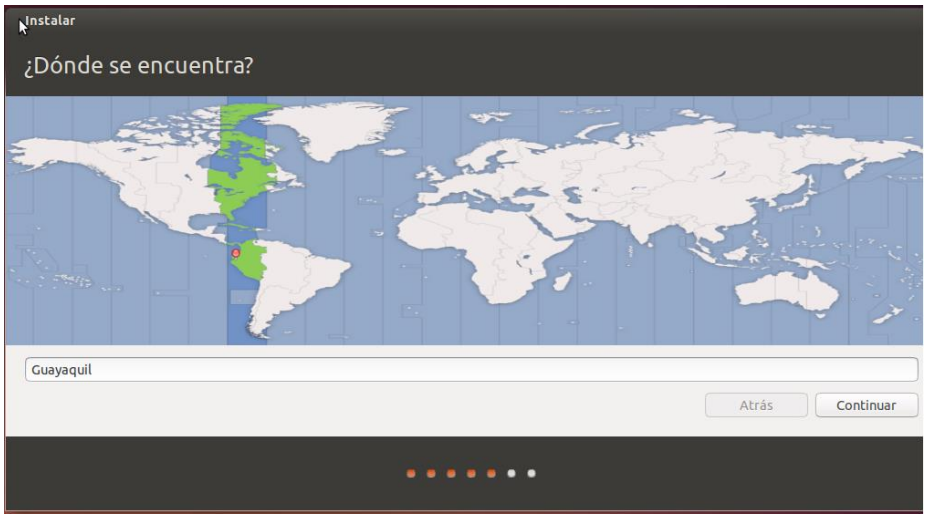


Figura 140. Instalación Ubuntu paso 5.

Elegimos la distribución de teclado en mi caso será español latinoamericano y seleccionamos la opción continuar.

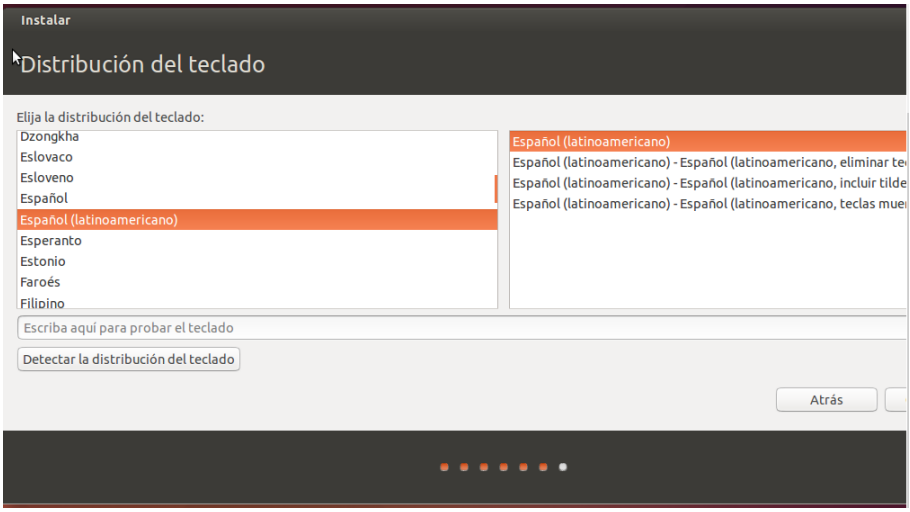


Figura 141. Instalación Ubuntu paso 6.

Asignamos un nombre y contraseña al equipo y se debe seleccionar la opción continuar.

Figura 142. Instalación Ubuntu paso 7.

Al terminar iniciará la instalación normal de Ubuntu.

Figura 143. Instalación Ubuntu paso 8.

Al finalizar con la instalación se debe iniciar sesión para continuar.

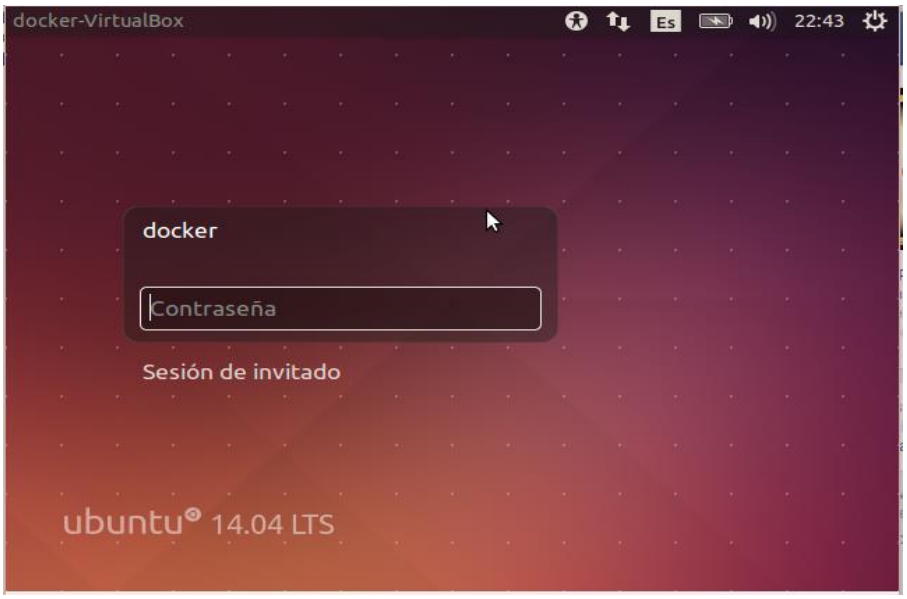


Figura 144. Ingreso al sistema operativo Ubuntu.

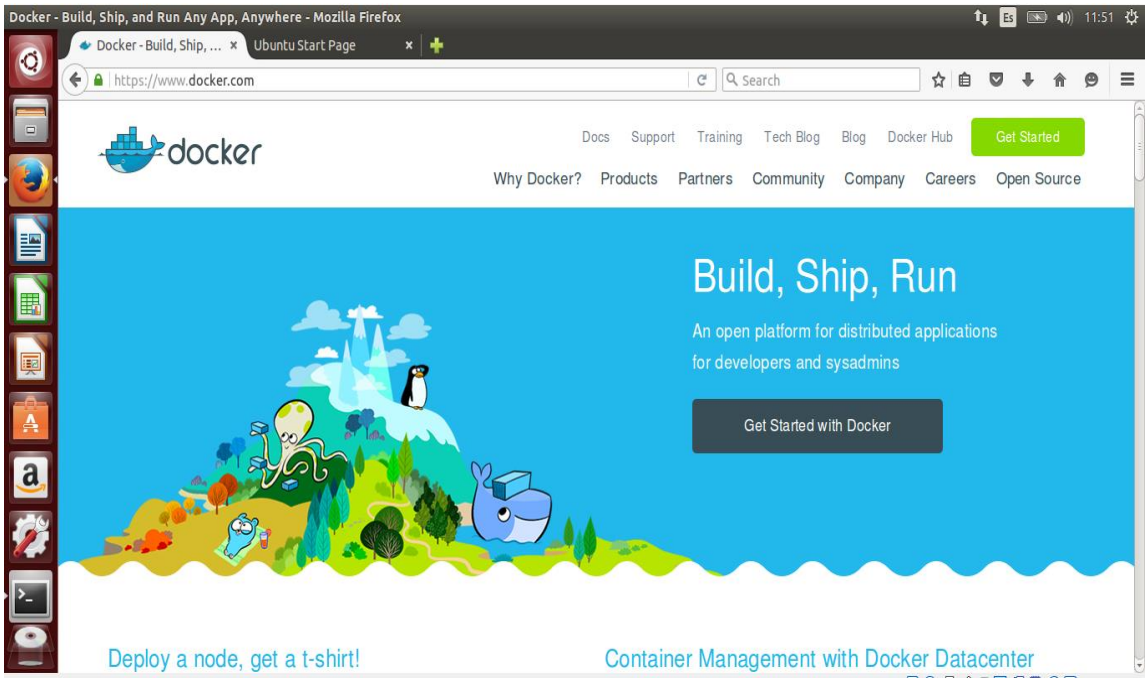


Figura 145. Página oficial Docker en Ubuntu 14.04.

- **Página oficial de Docker.**

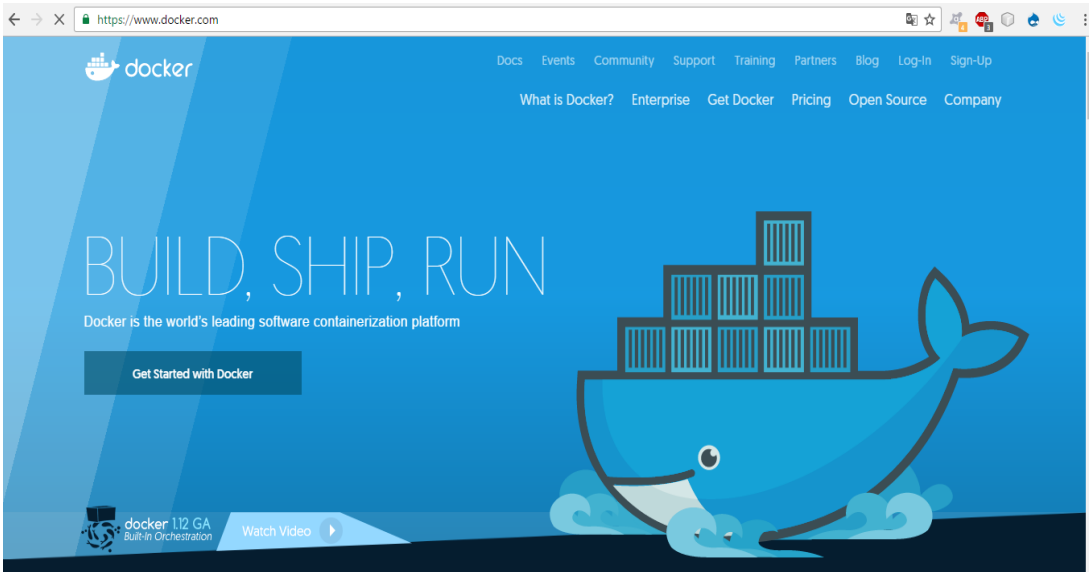


*Figura 146.*Página oficial de docker.

- **Página oficial Bitnami.**

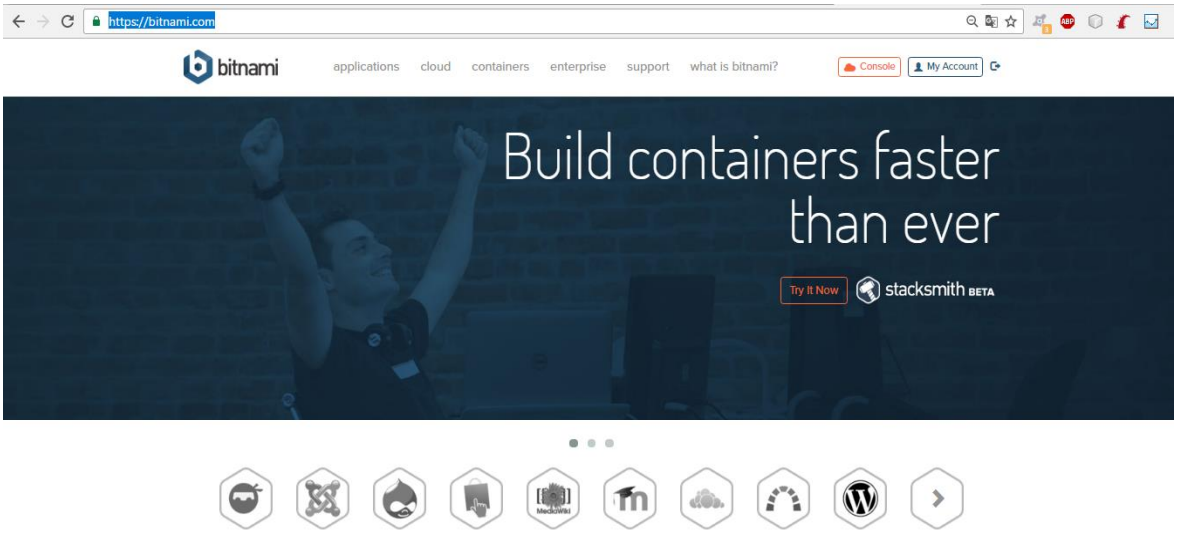


*Figura 147.*Página oficial de Bitnami.