https://doi.org/10.22201/dgtic.ctud.2023.1.1.12

Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 9 - 16

Implementación del ambiente de desarrollo de un sistema mediante máquinas virtuales

Información del reporte:

Para citar este reporte técnico:

Hernández Mayorga, M. A. (2023). Implementación del ambiente de desarrollo de un sistema mediante máquinas virtuales. *Cuadernos Técnicos Universitarios de la DGTIC, 1* (1), páginas (9 - 16). https://doi.org/10.22201/dgtic.ctud.2023.1.1.12

Mario Alberto Hernández Mayorga

Dirección General de Cómputo y de Tecnologías de Información y Comunicación

Universidad Nacional Autónoma de México

mariohm@unam.mx

ORCID: 0009-0002-5504-6009

Resumen:

Se describe cómo se utilizó la herramienta *Vagrant* para automatizar la creación de los ambientes de trabajo y el proceso realizado para replicar la implementación del ambiente de desarrollo mediante máquinas virtuales para la segunda versión del Sistema Integral de Personal Académico en la Dirección General de Cómputo y de Tecnologías de Información y Comunicación, así como los beneficios obtenidos, tales como facilitar la incorporación de miembros al equipo de desarrollo y la gestión de los ambientes de desarrollo, integración y pruebas.

Palabras clave:

Ambientes de desarrollo, máquinas virtuales.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.12 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 10 - 16

1. INTRODUCCIÓN

Un ambiente de desarrollo es un espacio de trabajo con un conjunto de procesos y herramientas usadas para programar, integrar, probar, validar y ejecutar un producto de *software* (Software-und System-Entwicklung, 2023). Estos ambientes facilitan el proceso de desarrollo, el control de versiones y la realización de pruebas, lo cual agiliza el flujo de trabajo. Históricamente, la forma de crear dichos ambientes ha implicado instalar y configurar manualmente los requerimientos técnicos del producto de software —base de datos, lenguaje de programación, bibliotecas o librerías, servidor web, entre otros— en la computadora local de cada uno de los integrantes del equipo de trabajo (Hashimoto, 2013).

Sin embargo, dado que cada proyecto tiene requerimientos específicos que pueden ser incompatibles con los de otros, entre ellos, la versión de *PHP* o la configuración del servidor web, la instalación de dichos requerimientos para varios proyectos a la vez en una máquina local puede convertirse en una tarea difícil de lograr. Además, se puede incrementar la dificultad del proceso si se usan diferentes sistemas operativos en las computadoras locales, ya que puede suceder que alguno de los requerimientos o una versión en específico no exista para alguno, o bien que necesite configuraciones particulares.

Por otra parte, la virtualización de los ambientes de trabajo permite construir ambientes independientes para cada proyecto, por lo cual cada uno puede contar con su propio servidor web, manejador de base de datos y versión del lenguaje de programación, así como cualquier otra dependencia, además de que es posible simular el ambiente de producción (Peacock, 2015).

Con base en lo anterior, en este reporte técnico se describe el contexto de la implementación de los ambientes de trabajo para la segunda versión del Sistema Integral de Personal Académico y el proceso realizado para automatizar, mediante máquinas virtuales, su creación. Dicha automatización facilitó la gestión de los ambientes de desarrollo, integración y pruebas, así como la incorporación de miembros al equipo de desarrollo.

2. OBJETIVO

Automatizar la creación de ambientes de trabajo (desarrollo, integración y pruebas) para la segunda versión del Sistema Integral de Personal Académico con el fin de proveer un mecanismo de replicación del ambiente de desarrollo.

3. DESARROLLO

3.1 CONTEXTO

Parte importante de la gestión de la información del personal académico de la Dirección General de Cómputo y de Tecnologías de Información y Comunicación (DGTIC) de la Universidad Nacional Autónoma de México (UNAM) es la creación, evaluación, revisión y firma de informes y programas de trabajo. El Sistema Integral del Personal Académico (SIPA) permite la gestión de dicha información.

La primera versión del SIPA fue responsabilidad de una sola persona que gestionó todo el proceso de implementación del sistema, el cual contó con un ambiente de desarrollo local. Debido a que no había

https://doi.org/10.22201/dgtic.ctud.2023.1.1.12 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 11 - 16

más personas involucradas en el proyecto, la gestión de los ambientes de trabajo no representaba un problema. Para el desarrollo de la segunda versión, la integración de un equipo de trabajo generó nuevas condiciones a considerar, entre ellas:

- La incorporación de miembros de diferentes perfiles (desarrollo y diseño gráfico).
- La necesidad de compartir código fuente (control de versiones).
- El acondicionamiento de un ambiente de desarrollo homogéneo, con los requerimientos técnicos del sistema; para este caso: Sistema Operativo *Linux*, Lenguaje de programación *PHP* (*framework Symfony*), base de datos *MariaDB* y servidor web *Apache*.
- La creación de ambientes de trabajo para desarrollo y pruebas.

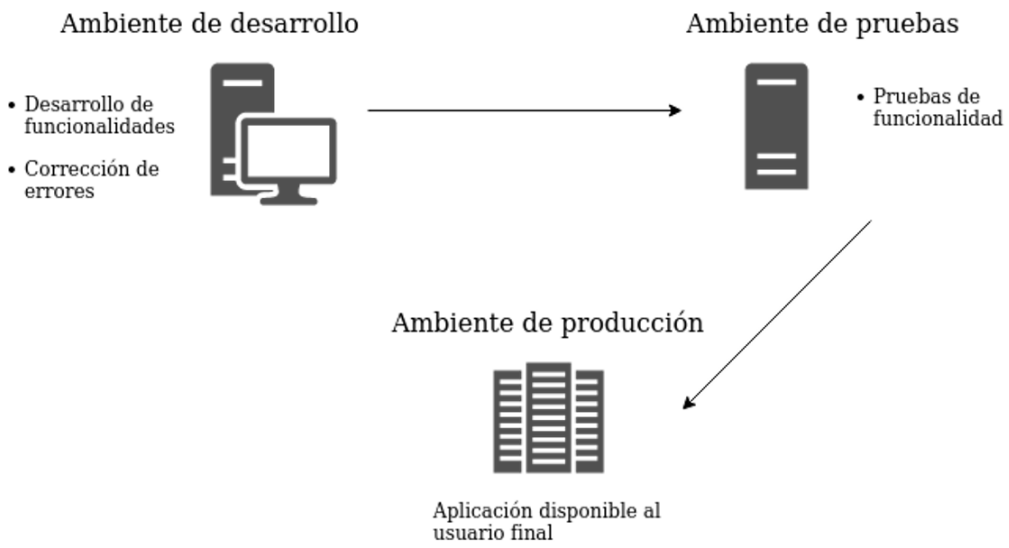
Cabe señalar que la duración del proyecto se definió a partir de los plazos establecidos para la entrega de la documentación del personal académico, por lo que la optimización del tiempo se convirtió en una prioridad. En este sentido, para poder comenzar a codificar, era necesario que los requerimientos técnicos del sistema estuvieran instalados y configurados en las computadoras de cada integrante del equipo, lo que podía ser una tarea repetitiva y que consumiría tiempo.

También se necesitaba instalar una versión del sistema donde el equipo de pruebas pudiera validar su correcto funcionamiento y reportar errores, para ello se necesitaba una nueva instalación y configuración de las herramientas para ejecutar una instancia del sistema.

Finalmente, también era necesario instalar los requerimientos en el servidor de producción, para liberar las versiones estables del sistema. En la figura 1 se muestran los ambientes de trabajo identificados.

Figura 1

Ambientes de trabajo



https://doi.org/10.22201/dgtic.ctud.2023.1.1.12 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 12 - 16

3.2 VIRTUALIZACIÓN DE LOS AMBIENTES DE TRABAJO

Es importante gestionar los ambientes de trabajo para desarrollo, pruebas y producción, y mantenerlos separados, para que se puedan realizar las tareas correspondientes en cada uno de ellos, sin afectar a los demás (Chamú Arias & González Guízar, 2020), propósito que se cumple a través de la virtualización de los ambientes de trabajo, lo que adicionalmente permite obtener los siguientes beneficios:

- Evitar la instalación y configuración de requerimientos directamente en la máquina local de cada uno de los integrantes del equipo.
- Replicar el ambiente de desarrollo de forma consistente.
- Emular la configuración del servidor de producción.
- Minimizar errores al desplegar el sistema en el servidor de producción.

En general se encontraron dos formas de gestionar los ambientes: con máquinas virtuales y con contenedores. Una máquina virtual es una representación virtual o emulación de una computadora física. Un contenedor es un paquete de aplicación que contiene todos los elementos necesarios para ejecutarse.

A continuación se muestra una tabla comparativa de las principales diferencias entre contenedores y máquinas virtuales:

Tabla 1

Tabla comparativa entre contenedores y máquinas virtuales

Característica	Contenedores	Máquinas virtuales
Sistema Operativo	Compartido con el del equipo donde se ejecutan	Propio
Velocidad	Inicio y apagado en segundos	Inicio y apagado en minutos
Uso de recursos	Menor	Mayor
Tamaño	MB	GB
Aislamiento	Aislamiento a nivel de proceso.	Aislamiento completo del equipo donde se ejecuta.

Nota. Adaptada de Containers vs VMs (virtual machines): What are the differences? de Google Cloud, s.f., https://cloud.google.com/discover/containers-vs-vms.

Derivado del análisis de las dos formas de virtualizar descritas previamente, el uso de contenedores se presenta como la opción más ligera para virtualizar el ambiente de trabajo. Por ello, se exploró la opción de implementar el ambiente de desarrollo mediante contenedores, y se usó *Docker* que es la herramienta más popular y ampliamente usada para su manejo.

Durante su instalación se presentaron dificultades que comenzaron a consumir tiempo, la principal fue que las computadoras del equipo de trabajo tenían distintas características y sistemas operativos y no todas contaban con los requerimientos mínimos para la instalación de la herramienta. Se intentó instalar

https://doi.org/10.22201/dgtic.ctud.2023.1.1.12 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 13 - 16

Docker mediante el proyecto *Docker Toolbox*[1] (ya obsoleto), pero se comenzó a invertir más tiempo en la instalación y configuración particular de las computadoras. Además, se presentaron incompatibilidades en las versiones instaladas con *Docker Toolbox*.

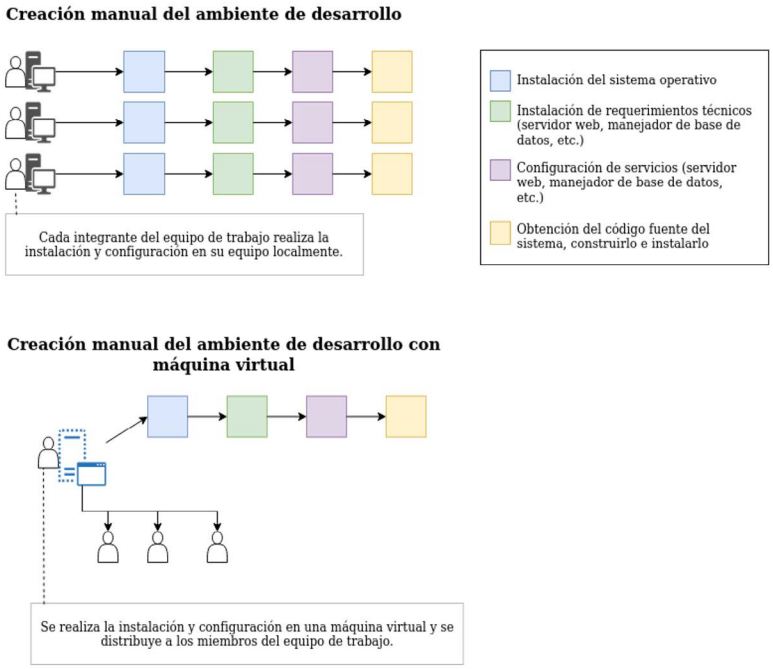
También se tomó en cuenta que el despliegue en producción del sistema, en ese momento, no se realizaba con contenedores, sino en un servidor virtual en el Centro de Datos de la DGTIC. Como dicho ambiente se simula de mejor manera con una máquina virtual, se optó por virtualizar los ambientes de trabajo a través de ella, considerando también la experiencia previa de algunos miembros del equipo en la materia.

3.3 AUTOMATIZACIÓN DE LA CREACIÓN DE LOS AMBIENTES DE TRABAJO

Como se mencionó en la introducción, históricamente, la forma de crear los ambientes de trabajo era mediante la instalación y configuración manual de los requerimientos técnicos del producto de *software*. Con máquinas virtuales, se realiza la instalación y configuración de una máquina virtual con los requerimientos necesarios para ejecutar el sistema, y después se distribuye a cada integrante del equipo de trabajo (figura 2). Así se tiene la certeza de que todos cuentan con las mismas versiones y configuraciones de las herramientas necesarias (Ramani, 2019).

Figura 2

Creación manual vs Creación con máquina virtual



[1] *Docker Toolbox* es un proyecto que permite ejecutar *Docker* en versiones de sistemas operativos *Windows* o *MacOS* que no cumplen con los requerimientos de la herramienta.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.12 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 14 - 16

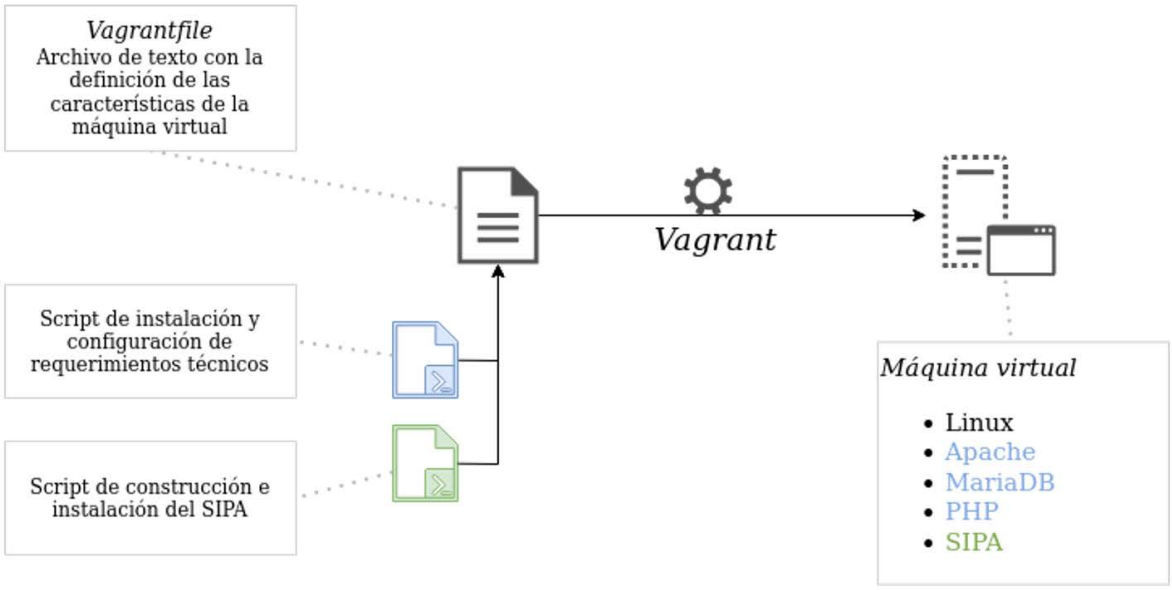
Sin embargo, un aspecto que se presentó fue que el tamaño del archivo que contiene la máquina virtual es grande (del orden de GB), por lo que su distribución a los integrantes del equipo se volvió tardada y aún era necesario instalar manualmente los requerimientos técnicos, aunque fuera una sola vez. Para atender estas situaciones, se buscaron herramientas de automatización, dentro de las cuales se encontró *Vagrant* como una buena opción, ya que es una herramienta para construir y administrar entornos de máquinas virtuales en un solo flujo de trabajo, fácil de usar y con un enfoque hacia la automatización.

La creación de la máquina virtual con *Vagrant* se realiza por medio de un archivo de texto, llamado *Vagrantfile,* que contiene la información necesaria para crear y replicar el ambiente. Además, *Vagrant* brinda opciones para aprovisionar la máquina virtual, desde *scripts de shell* hasta sistemas de administración de configuración.

Para aprovechar esta característica de la herramienta, para el desarrollo del SIPA, se crearon dos *scripts de shell,* uno con el fin de instalar los requerimientos del sistema, y el segundo, para la construcción e instalación de éste; los *scripts* se reutilizaron para instalar el sistema en producción (figura 3).

Figura 3

Uso de Vagrant para la creación automatizada de la máquina virtual

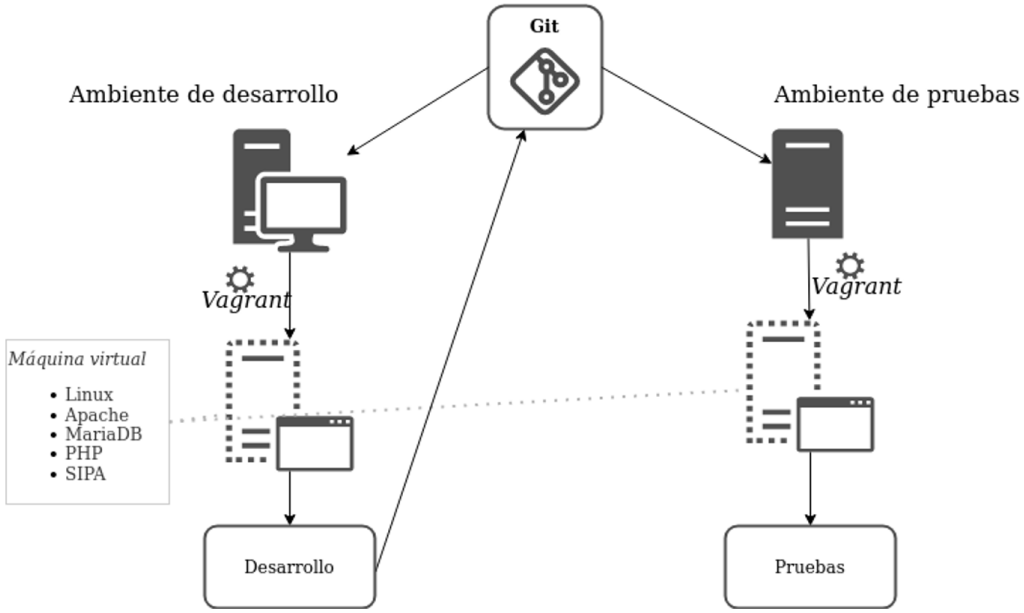


Finalmente, el archivo de texto para generar la máquina virtual y los *scripts* para configurar los requerimientos técnicos se agregaron al sistema de control de versiones del proyecto, permitiendo que las actualizaciones al ambiente de desarrollo también pudieran tener control de cambios. De esta forma, al obtener o actualizar el código fuente en el sistema de control de versiones, también se incluyen los archivos necesarios para crear la máquina virtual con el ambiente de desarrollo local, y así, comenzar el desarrollo por parte de cada uno de los miembros del equipo, en cuestión de minutos. Los ambientes para las etapas de integración y pruebas se crearon de la misma forma (figura 4).

https://doi.org/10.22201/dgtic.ctud.2023.1.1.12 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 15 - 16

Figura 4

Creación de ambientes de desarrollo y pruebas



4. RESULTADOS

Se logró automatizar la creación de ambientes de trabajo con máquinas virtuales para el desarrollo del SIPA, lo cual permitió:

- Replicar el ambiente de desarrollo de forma consistente para cada uno de los integrantes del equipo (desarrollo y diseño gráfico).
- Reducir el tiempo de creación y configuración del ambiente de desarrollo en aproximadamente un 70 por ciento.
- Crear los ambientes necesarios para las etapas de integración y pruebas.
- Reutilizar el *script* de instalación del sistema en el servidor de producción.

Como trabajo a futuro, se propone explorar la posibilidad de usar tecnologías como *Puppet o Chef* para la instalación de los servicios o requisitos técnicos en la máquina virtual en lugar de *scripts* de *shell*. Dichos programas permiten definir los servicios a instalar y configurar el servidor de forma transparente.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.12 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 16 - 16

5. CONCLUSIONES

Automatizar la creación de los ambientes de trabajo para cada etapa (desarrollo, integración y pruebas) permite realizar las actividades de cada una de ellas en un ambiente aislado y permite que el flujo de trabajo se ejecute de forma más ágil.

Se identificó que la virtualización de ambientes de desarrollo con máquinas virtuales puede ser una alternativa cuando por el contexto de un proyecto no es posible llevarlo a cabo con contenedores.

El uso de *Vagrant* para la creación de ambientes en máquinas virtuales facilitó la gestión de los ambientes de desarrollo, integración y pruebas, permitió la replicación consistente de los ambientes, y redujo el tiempo de acondicionamiento del ambiente de desarrollo así como la distribución del mismo en el control de versiones del proyecto.

REFERENCIAS BIBLIOGRÁFICAS

Chamú Arias, J. O., y González Guízar, A. (2020). *La importancia de separar los ambientes de trabajo en el desarrollo de los productos de software*. Boletín Informativo para el Personal de la DGTIC, 13. https://www.tic.unam.mx/boletin-13/

Google Cloud. (s.f.). *Containers vs VMs (virtual machines): What are the differences?* Temas Google Cloud. https://cloud.google.com/discover/containers-vs-vms

Hashimoto, M. (2013). *Vagrant: Up and running*. (1ra. ed). O'Reilly Media, Inc.

Peacock, M. (2015). *Creating development environments with vagrant: Leverage the power of Vagrant to create and manage virtual development environments with Puppet, Chef, and VirtualBox*. (2a. Ed). Packt Publishing.

Ramani, M. (2019). *Moving towards devops with Vagrant*. https://education.dell.com/content/dam/dell-emc/documents/en-us/2019KS_Ramani_Moving_Towards_DevOps_with_Vagrant.pdf

Software- und System-Entwicklung (2023). Development Environment. *Technology Definitions | SUSE Defines*. https://www.suse.com/suse-defines/definition/development-environment/