

**DESPLIEGUE AUTOMÁTICO DE
INFRAESTRUCTURAS HPC MULTINÚCLEO Y
MULTIMÁQUINA CON MÁQUINAS VIRTUALES
UTILIZANDO VAGRANT**

AUTOR:

HÉCTOR FABIO JIMÉNEZ SALDARRIAGA

**UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA**

2020

**DESPLIEGUE AUTOMÁTICO DE
INFRAESTRUCTURAS HPC MULTINÚCLEO Y
MULTIMÁQUINA CON MÁQUINAS VIRTUALES
UTILIZANDO VAGRANT**

AUTOR:

HÉCTOR FABIO JIMÉNEZ SALDARRIAGA

**Documento con propuesta de proyecto de grado presentado
como requisito para optar al título de Ingeniero de Sistemas
y Computación**

DIRECTOR:

**Ramiro Andrés Barrios Valencia
MSc. en Ingeniería de Sistemas y Computación
Especialista en Electrónica Digital
Grupo de Investigación Sirius**

**UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA**

2020

Dedicatoria

A Dios primeramente por permitirme estar en el buen camino y finalizar este gran paso en mi educación, a mis padres quienes me dieron todo su apoyo, amor infinito y fortaleza en todos los momentos mas difíciles de mi carrera. A mis hermanas, Leidy, Ana, y Cindy quienes son el motor para ser mi mejor versión cada día. A mi novia, Melissa quien siempre me apoyo incondicionalmente.

Héctor Fabio Jiménez Saldarriaga

Agradecimientos

Al ingeniero Ramiro Andrés Barrios Valencia por todo el apoyo brindado y las contribuciones realizadas durante el desarrollo de este proyecto de grado. A el grupo de Investigación Sirius por permitirme el uso de todos los recursos computacionales para experimentar, además de haberme brindado la posibilidad de conocer grandes personas.

Finalmente a la Universidad Tecnológica de Pereira por los conocimientos, experiencias durante la trayectoria de carrera, con los cuales fue posible la realización de este proyecto.

Contenido

1. Título	9
2. Introducción	9
3. Formulación del Problema	10
4. Justificación	11
5. Objetivos	12
5.1. Objetivo general	12
5.2. Objetivos específicos	12
6. Marco Referencial	12
6.1. Marco Histórico	12
6.2. Marco Conceptual	13
6.3. Marco Teórico	15
7. Alcance	17
8. Diseño Metodológico	17
8.1. Metodología	19
9. Implementación	20
9.1. Configuraciones Iniciales y Prerrequisitos	20
9.1.1. Instalación de Oracle VM VirtualBox	20
9.1.2. Instalación de Vagrant	24
9.2. Pruebas de Aprovisionamiento Vagrant Boxes	30

9.3. Pruebas de Networking y Direcccionamiento	34
9.3.1. Opciones de Configuración Programáticas Vagrant	37
9.4. Aproveccionamiento de Múltiples Maquinas Virtuales	39
9.5. Sistema de Archivos Compartido clúster MPI	40
9.6. Configuraciones de Seguridad	44
9.7. Pruebas de clúster MPI Multimaquina	45
10.Resultados	50
11.Participantes	52
12.Recursos y Presupuesto	52
12.1. Recursos Humanos	52
12.2. Recursos Físicos	52
13.Cronograma	53
14.Conclusiones	55
15.Anexo A: VagrantFile Cluster HPC MPI, Multinucleo Multimaquina	56
16.Bibliografía	57

Lista de Figuras

1.	Sitio Web Oficial, https://www.virtualbox.org/	21
2.	Instalador Virtualbox, Versión 6.1.6	21
3.	Ejecución de Wizard como administrador.	22
4.	Es obligatorio instalar VirtualBox con las opciones resaltadas, soportando los dos modos de Interfaz de red.	22
5.	Se permite crear accesos directos y registro de extensiones preferencia, Wizard.	23
6.	Características de Networking y Driver requieren reinicio de la máquina host.	23
7.	Instalación completa VirtualBox.	24
8.	Sitio web oficial de vagrant, sección de descarga	25
9.	Instalador de Vagrant, setup wizard.	26
10.	Setup Wizard de Vagrant.	26
11.	Selección de ubicación para instalar Vagrant. La recomendación oficial es dejar en la ruta por defecto, ya que de esta forma no habrá conflicto para definirlo dentro de las variables de entorno (la definición dentro de las variables de entorno es realizado por Vagrant de manera automática).	27
12.	Paso de confirmación para el install wizard	28
13.	Instalación de Vagrant en proceso.	28
14.	Opciones disponibles de Vagrant en la línea de comandos.	29
15.	Flujo de Trabajo de Vagrant, Etapas.	31
16.	Ejemplo de Vagrantfile, versión 1.	32
17.	Ejemplo de Vagrantfile, versión 2.	32
18.	resultado de importación y creación de vagrant up.	33

19.	Aprovisionamiento Exitoso, Estado de la máquina virtual.	33
20.	Conexión exitosa a la máquina virtual aprovisionada, vagrant ssh . .	34
21.	Sección de red, máquina virtual aprovisionada.	35
22.	Conexión a la máquina virtual aprovisionada, vagrant ssh	36
23.	Conexión usando el modo Host Only, Implementación del Cluster MPI, cada máquina virtual Guest queda en el mismo segmento de red permitiendo comunicación bidireccional.	36
24.	Conexión a la máquina virtual aprovisionada, vagrant ssh	37
25.	Port Forwarding	38
26.	Multiple Port Forwarding, con puertos diferentes en máquina host e invitado, especificando protocolo.	38
27.	Declaración de red privada, asignación de IP por dhcp.	38
28.	Declaración de red privada, asignación de dirección ipv4 estática. . . .	39
29.	Defunción de Múltiples máquinas virtuales, vagrant ssh	39
30.	Conexión a la máquina virtual aprovisionada, vagrant ssh	40
31.	Funcionamiento de NFS.	41
32.	Funcionamiento de NFS, ejemplo basico NFSv3.	42
33.	Instalación Servidor NFSv4.	42
34.	Configuración NFSv4, Clientes del Cluster.	43
35.	Arquitectura funcional cluster MPI, virtual switch	45
36.	StrictHostChecking, Conexiones SSH deshabilitado	46
37.	Interbloqueo de procesos, Fallo de Instalación de Servicio NFS	46
38.	Solución Programática.Se Deshabilita Servicio problemático.	47
39.	Lanzamiento de Cluster estado de los nodos no creados.	47
40.	Lanzamiento de Cluster en Marcha.Todos los nodos se encuentran en ejecución.	47

41.	Lanzamiento de Cluster en Marcha.Prueba basica de conectividad maquina master-vm.	48
42.	Lanzamiento de Cluster en Marcha.Prueba de Conectividad incluyendo a todas las maquinas del cluster.	48
43.	Lanzamiento de Cluster en Marcha.Pruebas con time.	48
44.	Destruccion de entorno de Cluster.	49

Lista de Tablas

1. Título

"Despliegue Automático de Infraestructuras HPC multinúcleo y multimáquina con máquinas virtuales utilizando Vagrant"

2. Introducción

La computación de alto rendimiento o HPC de sus siglas en inglés *High Performance Computing* es un campo especial e importante de las ciencias de la computación que implica resolver problemas complejos que demandan grandes capacidades de recursos computacionales⁽¹⁾(*como ejemplo un gran de número de procesadores, alta cantidad de Memoria, alta capacidad de almacenamiento*), es una de las mejores maneras de computar algoritmos en diferentes campos de la ciencia como la física, medicina, bioinformática, mecánica, química u otros problemas comunes de Big Data. Estas tareas y procesos son grandes para ser realizados en una sola máquina, por ello se hace necesario construir lo que se denomina un *Clúster* de computadores definido como un grupo de múltiples nodos o computadores que trabajan de manera conjunta y distribuida para cumplir con una meta común.

Debido a la naturaleza de múltiples computadores conectados entre sí, existen diferentes modelos de computación paralela y distribuida que han sido desarrollados e implementados por la comunidad científica para la coordinación de los computadores pertenecientes a un Clúster, de manera que su uso sea eficiente y efectivo. Un modelo muy utilizado y popular es el *paso de mensajes*, una técnica empleada en programación concurrente para aportar sincronización entre procesos y permitir la exclusión mutua; su diseño esta pensando para algoritmos y programas que exploten la existencia de múltiples procesadores en un Clúster. MPI es el estándar de facto preferido por la comunidad HPC a la hora de realizar implementaciones de paso de mensajes.

Tradicionalmente, configurar un clúster de computadores, por ejemplo, un clúster MPI, es una tarea desafiante que requiere que estudiantes, aficionados, novatos y avanzados pasen un tiempo configurando los diferentes elementos del sistema operativo, configuraciones de red, configuraciones de aplicaciones terceras. Sin embargo, en los últimos años el avance del Cloud Computing y la implementación de metodologías ágiles que impulsan culturas como la DevOps han hecho que la automatización en todas las etapas sea masiva [11] [10] , gracias a esto diferentes

proyectos de software libre y de código abierto han visto la luz, como Vagrant, una herramienta de código abierto que permite la creación y configuración de ambientes virtuales portables y ligeros. En este proyecto se presenta una implementación utilizando Vagrant para realizar la construcción automática de un clúster MPI multinúcleo y multimáquina.

3. Formulación del Problema

La infraestructura como código o IaC (Infrastructure as Code) es uno de los pilares fundamentales de la cultura DevOps, la IaC hace referencia a la práctica de poder administrar, aprovisionar, actualizar, monitorear y configurar la infraestructura TI de las empresas, utilizando scripts en diferentes lenguajes de programación en lugar de configurar las máquinas de manera manual [21] [23]. La IaC trata todas las configuraciones de la infraestructura TI como código, permite a las máquinas virtuales gestionarse de manera programada y automática, lo que elimina la necesidad de realizar configuraciones manuales por parte del personal a cargo de componentes individuales de hardware. Esto hace que la infraestructura sea muy moldeable, es decir, escalable y replicable siendo agnóstico al hardware y al proveedor de servidor cloud, además se utilizan las mismas prácticas implementadas para la gestión y versionamiento del código de aplicaciones de software.

La implementación y construcción de un clúster MPI implica tiempo y conocimiento de todos los elementos que lo componen, es necesario conocer un completo detalle de como realizar la instalación de cada uno de los componentes del clúster, algunas tareas que son repetitivas se puede automatizar implementando IaC; ejemplo de ello es la creación de usuarios con sus niveles de privilegios, las configuraciones de seguridad, la configuración de servicios, las configuraciones de sistemas de archivos compartidos, entre otras tareas repetitivas. Es necesario en un clúster MPI asegurar la inmutabilidad y homogeneidad en las configuraciones realizadas en los nodos que lo componen. [24]

4. Justificación

En la universidad Tecnológica de Pereira, el programa de Ingeniería de Sistemas y Computación brinda la posibilidad a los estudiantes de pregrado tomar como materia electiva de noveno semestre Computación de Alto Desempeño(*CAD*) o HPC(*High Performance Computing*), en esta materia el estudiante tiene la posibilidad de aprender y conocer todos los elementos que implica la computación de alto desempeño, además el aprendizaje a tratar grandes volúmenes de datos, ejecutar grandes y complejos cálculos científicos, en tiempos aceptables, medibles, con una precisión adecuada a los diferentes problemas establecidos por el docente. Los estudiantes también aprenden a construir un mini clúster replica con nodo maestro y múltiples nodos esclavos, esta simulación permite que el estudiante se haga a una idea de las diferentes arquitecturas paralelas y distribuidas encontradas en un clúster real, así como los diferentes modelos de memoria. La construcción y montaje de este es realizada por los estudiantes con la guía del docente.

Durante el segundo semestre del año 2018 en el curso de Computación de Alto Desempeño, se tuvo la oportunidad de construir el clúster: el montaje, aprovisionamiento y puesta en marcha. Durante las múltiples revisiones se observaba que se presentaban algunas fallas como :

- Falta de configuraciones de networking en las máquinas esclavas.
- Falta de idempotencia y definiciones específicas para las configuraciones de las máquinas virtuales.
- Fallos de arranque con las máquinas virtuales.
- Pérdida de estados de las máquinas esclavas.

Es así como los puntos anteriores generaron cuestionamientos y reflexiones acerca de la forma de automatizar el proceso de la construcción del clúster de HPC utilizando tendencias modernas como IaC (*Infrastructure as Code*) e implementación de herramientas de la cultura DevOps como **Vagrant** y **Git**. En este proyecto se presenta una implementación que automatiza la construcción automática de un clúster MPI multinúcleo y multimáquina, de esta manera el estudiante podrá poner su atención principalmente en los conceptos de modelos de algoritmos concurrentes, paradigmas y modelos de la programación paralela e implementación de la programación de memoria distribuida y paso de mensajes con MPI.

5. Objetivos

5.1. Objetivo general

Diseñar e implementar un conjunto de programas para el aprovisionamiento automático de un clúster computacional sobre máquinas virtuales, que soporten procesos con MPI

5.2. Objetivos específicos

- Obtener la lista de requerimientos, limitaciones, especificaciones para el diseño del aprovisionamiento automático utilizando Vagrant.
- Desarrollar e implementar un sistema de archivos distribuido en el Clúster MPI.
- Realizar pruebas de ejecución y funcionamiento del clúster.
- Realizar pruebas de ejecución y desempeño multimáquina en el Clúster MPI compuesto por máquinas virtuales.

6. Marco Referencial

6.1. Marco Histórico

La historia de la computación de alto desempeño tiene su origen en los conceptos informáticos que se remontan a 2400 a. C. con la creación del ábaco. La informática electrónica comenzó a mediados de la década de 1940 con la ENIAC, la primera computadora electrónica de uso general [38].

« La primera asociación de High-Performance Computing (HPC) comenzó en 1985 cuando la National Science Foundation estableció una asociación entre cinco centros de investigación: San Diego Supercomputer Center (SDSC) en la Universidad de California en San Diego, Pittsburgh Supercomputer Center (PSC) en la Universidad de Pittsburgh, el Centro Nacional de Aplicaciones de Supercomputación (NCSA) en la Universidad de Illinois Champagne-Urbana, el Centro de Teoría Cornell en

la Universidad de Cornell y el Centro John von Neumann en la Universidad de Princeton.

»En 1997, la NSF anunció dos nuevas asociaciones de supercomputación, la Asociación Nacional para la Infraestructura Computacional Avanzada (NPACI) y la Alianza Nacional de Ciencia Computacional (NCSA). La financiación de estas asociaciones se limitó a cinco años.

»La siguiente iteración de las asociaciones de HPC fue TeraGrid, establecida en 2001. La financiación de TeraGrid también se limitó a cinco años. Se proporcionó una extensión de sexto año para cerrar la brecha entre TeraGrid y la asociación XSEDE, que recibió fondos en 2011. La financiación fue por cinco años, hasta 2016, momento en el que se estableció XSEDE 2» [38] .

En cuanto a la cultura DevOps, se considera a Andrew Clay su creador, junto con el belga Patrick Debois. Ambos empezaron a trabajar en 2008 para llevar la filosofía Agile al mundo de la administración de sistemas. El resultado de su trabajo se haría público en San José (California) en 2009 y un año más tarde en Europa [5] .

6.2. Marco Conceptual

A continuación se describen los conceptos fundamentales para el entendimiento del presente proyecto

- *Gestor de Paquetes*: Un gestor de paquetes es una pieza de software que permite instalar, obtener, eliminar, actualizar software y sus dependencia en un sistema operativo, como se menciona en [34], la idea principal de esto es tener un control total de las librerías y dependencias que son requeridas por un software; mediante la implementación de una heurística se tiene una traza de todos las librerías instaladas en el sistema de archivos. El sistema gestor de paquetes logra esto gracias a la construcción de un árbol de dependencias, que se encuentra definido en un archivo de configuración dentro de los paquetes(*por ejemplo RedHat, tiene un sistema de empaquetamiento, las extensiones **rpm**, debian con los archivos **deb***), estos contienen una definición exacta de las ubicaciones donde se instalarán las dependencias según el estándar lsb de las distribuciones Linux, además de detectar todos los conflictos posibles mediante SAT solvers [34].

- *Multithreading*: En arquitectura de computadores, Multithreading o multihilo es la habilidad de una CPU (Unidad Central de Procesamiento) o de un core en un procesador multi-core, de proveer múltiples hilos de ejecución concurrentes, siendo esto soportado por el sistema operativo. En un sistema multihilo, los hilos comparten los recursos de un solo core o de múltiples cores, lo que incluye las unidades de computación, las caches de CPU y el TLB (Translation Lookaside Buffer) [31]. El multithreading busca incrementar el uso de un solo core mediante la utilización, tanto de paralelismo a nivel de hilo, como de paralelismo a nivel de instrucción [16].
- *Multiproceso*: El Multiprocesamiento o multiproceso es el uso de dos o más procesadores (CPU) en una computadora para la ejecución de uno o varios procesos de manera concurrente. Las ventajas principales son: el incremento del rendimiento, ya que al incrementarse el número de procesadores, se puede realizar mayor cantidad de trabajo en un menor tiempo; más económico al escalarse, al compararse con sistemas de monoprocesamiento, ya que es posible compartir periféricos, almacenamiento, suministro de energía...; y mayor fiabilidad, ya que al tener más de un procesador las funciones pueden ser distribuidas, y en el caso de que ocurra un fallo en un procesador, el sistema no se detiene sino que lo hace más lento [35].
- *Control de versiones*: Es un componente de la administración de configuraciones de software, también es conocido como control de revisiones, control de código fuente, o administración de código fuente. Se define como la gestión de cambios que se realiza de diferentes elementos o configuraciones en productos de software y sistemas informáticos. Gracias a esta necesidad surgieron los sistemas de control de versiones. Entre ellos se puede encontrar :
 - Git
 - Subversión
 - Mercurial
- *Git*: Git es un sistema de control de versiones distribuido, creado por Linus Torvalds el 7 de abril de 2005 para poder mejorar el flujo de trabajo de los desarrolladores del kernel de Linux, basado en algunas ideas de Bitkeeper. Su desarrollo fue potenciado por la comunidad debido al rompimiento comercial de Bitkeeper como compañía, su primera puesta en escena fue el manejo de cambios del código fuente del kernel Linux que cuenta con cientos y miles de desarrolladores alrededor de todo el mundo. En la actualidad es el control de

versión distribuido mas utilizado y es una de las herramientas mas necesarias para cualquier desarrollador de software.

- *DevOps*: Developer Operations, una palabra que inicialmente surgió de una discusión entre Andrew Clay y Patrick Debois en el año 2008. Ambos desarrolladores experimentados les preocupaban los inconvenientes de la metodología ágil y querían encontrar algo mejor. La idea comenzó a extenderse lentamente después de realizar un evento DevOpsDays celebrado en Bélgica en 2009, la palabra se convirtió en una palabra de moda.

DevOps es una idea, que busca la combinación de filosofías, prácticas y herramientas culturales que aumenta la capacidad de una organización para entregar aplicaciones y servicios con gran valor, evolucionando y mejorando productos a un ritmo más rápido que las organizaciones que usan procesos tradicionales de desarrollo de software y gestión de infraestructura. Esta velocidad permite a las organizaciones servir mejor a sus clientes y competir de manera más efectiva en el mercado. [4]

La implementación de una cultura DevOps garantiza que los desarrolladores ahora puedan participar en los despliegues de aplicaciones a producción, los administradores pueden escribir scripts y los ingenieros de QA saben cómo resolver otros problemas además de las pruebas. Los procesos pueden automatizarse y nadie tiene que esperar, ya que ahora pueden trabajar más estrechamente y encontrar soluciones más rápidas y mejores. Esto permite que haya una mejor comunicación y comprensión entre los equipos de desarrollo y operaciones. [29]

- *Metodologías Ágiles*: Son aquellas metodologías que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo una respuesta flexible e inmediata ante las necesidades del desarrollo del proyecto y las circunstancias específicas del entorno [36]. Estas metodologías son especialmente útiles cuando se tiene que abordar un proyecto donde probablemente ocurrirán cambios, ya que proporcionan mecanismos para afrontar estos cambios manteniendo el máximo valor para el cliente, incrementando así la probabilidad de finalizar con éxito el proyecto [32] [1].

6.3. Marco Teórico

- *Hypervisor*: Un hypervisor, conocido también como monitor de máquina virtual (*Virtual Machine Manager*), es un software que crea y ejecuta máquinas

virtuales (VMs) y que, además, aísla el sistema operativo y los recursos del hypervisor de las máquinas virtuales. El rol importante de esta herramienta es permitir realizar todas las actividades de administración y gestión de las máquinas virtuales. [3]

- *máquina Virtual*: Una máquina virtual (VM del inglés Virtual Machine) es un entorno virtual que funciona como sistema informático virtual con su propia CPU, memoria, interfaz de red y almacenamiento, pero se crea en un sistema de hardware físico, en el que está instalado un hypervisor libre o comercial al que se denomina como máquina *host*, y las VMs que utilizan sus recursos se denominan *guest*. Las VMs se encuentran aisladas del resto del sistema, pero puede haber varias VMs en una sola pieza de hardware, como un servidor. [2]
- *Message Passage Interface*: Es el estándar que define la sintaxis, la semántica y demás elementos de la especificación MPI para una interfaz común de funciones implementadas para el paso de mensajes. En la actualidad MPI cuenta con un conjunto de funciones convergentes en una librería. El estándar MPI es construido a través de un proceso abierto y comunitario alrededor de todo el mundo, conformada por investigadores, científicos, desarrolladores de aplicaciones y múltiples vendedores que trabajan en problemas de computación de alto desempeño y paradigmas de procesamiento paralelo. Este estándar busca explotar el uso de la programación paralela y el paso de mensajes entre procesos, posibilitando utilizar todos los cores existentes en un procesador o en máquinas multiprocesador. Según el MPI-2019-Draft-report [?] [2] MPI no es un lenguaje de programación en sí, y todas las operaciones MPI son expresadas como funciones, subrutinas o métodos, que definen de manera clara todos los mecanismos para un **estándar** de paso de mensajes.
- *Network File System*: El sistema de archivos en red (Network File System, NFS) es una implementación cliente/servidor que permite a un usuario de un equipo ver, almacenar y actualizar archivos en un equipo remoto como si se estuviera en el equipo del usuario. Este protocolo es uno de varios estándares de sistemas de archivos distribuidos más implementado en dispositivos de almacenamiento (NAS). El protocolo NFS permite al usuario o administrador del sistema designar como accesible todo o una porción del sistema de archivos en un servidor. Esta parte puede ser accedida por los clientes con los privilegios que se asignen a cada archivo, sólo de lectura o lectura y escritura. NFS utiliza llamadas de procedimiento remoto (RPC) para enrutar solicitudes entre clientes y servidores [28]. Todos los sistemas Unix pueden trabajar con este

protocolo; si se involucran sistemas Windows, se debe utilizar Samba en su lugar [22]

7. Alcance

Este proyecto tiene como fin realizar la implementación de scripts que permitan automatizar la creación de un clúster HPC que permita realizar la ejecución de aplicaciones y algoritmos desarrollados utilizando MPI mediante el uso de máquinas virtuales automatizadas utilizando la herramienta Open Source **Vagrant**, una herramienta de línea de comandos que permite administrar el ciclo de vida de máquinas virtuales de manera consistente y maximizar la productividad soportando múltiples hypervisores y proveedores como VirtualBox, VMware, Hyper-v, AWS, GCP, entre otros.

Pensando en futuras implementaciones, se tiene la posibilidad de que a partir de los resultados que se obtuvieron en el presente proyecto, se podría implementar una configuración para otros proveedores de la nube como AWS o Google Cloud Platform.

8. Diseño Metodológico

- El enfoque y método seguido para la ejecución de este proyecto es “*aprender haciéndolo*”, parte desde el aprendizaje de las características de Vagrant y la programación de los scripts en Ruby hasta la parametrización y configuración de cada una de las tecnologías y software implicados para dejar el clúster de MPI funcionando, todo ello llevado a cabo desde un punto de vista práctico.
- Tipo de investigación

Basados en los puntos anteriores se considera que éste es un Proyecto de aplicación ya que se pretende desarrollar e implementar la automatización para la creación de un clúster hpc, buscando resolver problemas de la vida cotidiana y controlar situaciones prácticas, permitiendo la obtención de nuevos conocimientos por medio de la investigación realizada y la generación de un entregable después de la culminación de dicha investigación.

En esta investigación se utilizará un enfoque cualitativo, además, se empleará el método científico.

- Estrategias

Se llevará a cabo un conjunto de pruebas sobre el sistema, para establecer si cumple con las estimaciones del proyecto y si es funcional.

8.1. Metodología

Fases de la Metodología

- Fase 1, **Recolección de requerimientos y análisis de la información:** Durante esta etapa se generó un listado de actividades y herramientas opensource disponibles para ejecutar el proceso de automatización teniendo muy presente el curso de computación de alto desempeño y los pasos de creación de un cluster HPC utilizando el estándar MPI. También se realizó la búsqueda de la literatura recomendada sobre libros de Infraestructura como Código para poder llevar a cabo todo el proceso de programático y declarativo.
- Fase 2, **Diseño e implementación de despliegues automáticos:** Se realizaron pruebas experimentales de construcción manual de un cluster HPC utilizando VirtualBox para determinar los planes a ejecutar en nuestros archivos de programación declarativa. Se realizó la revisión documental de las diferentes herramientas utilizadas para el éxito de este proyecto.
- Fase 3, **Documentación y conclusiones:** Se realiza la creación de este documento, donde se detalla de manera clara y concisa la implementación realizada y cada una de sus etapas. Se realizan múltiples pruebas para realizar la validación de nuestra hipótesis y probar la idempotencia asegurada al usar vagrant.

9. Implementación

9.1. Configuraciones Iniciales y Prerrequisitos

Para realizar este proyecto de grado, las instalaciones de pruebas se implementaron utilizando el sistema operativo Windows 10 Professional version de 64 bits, junto con la herramienta de Windows Subsystem for Linux (WSL2).

El computador portátil utilizado para hacer el desarrollo de este trabajo de grado tiene las siguientes características de hardware :

- Marca : Lenovo
- Referencia : Thinkpad X1 6th
- Procesador : Intel(R) Core(TM) i7-8650U CPU @1.90GHz 2.11GHz
- Memoria Ram : 16GB
- Disco Duro : 1TB SSD
- Sistema Operativo : Windows 10 Professional, Build 19631.mn.release200514

A continuación se detallarán los elementos que deben ser instalados para poder llevar acabo todo el proyecto, además de algunos detalles de la implementación y apuntes sobre las herramientas.

9.1.1. Instalación de Oracle VM VirtualBox

VirtualBox es un Hypervisor opensource para procesadores Intel/AMD multiplataforma; al momento de realizar este documento la versión estable es la **6.1.6**. Su instalación se realiza mediante un wizard de manera guiada, éste instala los elementos necesarios para poder realizar la virtualización de todos los dispositivos necesarios por las máquinas virtuales, además de instalar drivers adicionales para montar pseudo dispositivos de red como se muestra en las siguientes imágenes. Actualmente existe la versión privativa Oracle VM VirtualBox, que es gratuita únicamente **bajo uso personal o de evaluación** como es el de este proyecto¹. Oracle Corporation adquirió Sun Microsystems en enero de 2010, es por esto que hubo un cambio de la marca del producto a *Oracle VM VirtualBox* [6].

¹(VirtualBox Personal Use and Evaluation License o PUEL)

Es necesario ingresar al sitio web de VirtualBox y realizar la descarga del instalador.



Figura 1: Sitio Web Oficial, <https://www.virtualbox.org/>.

Una vez terminada la descarga, es necesario realizar la validación de que fue correctamente descargado, esto se puede validar con una firma hash.

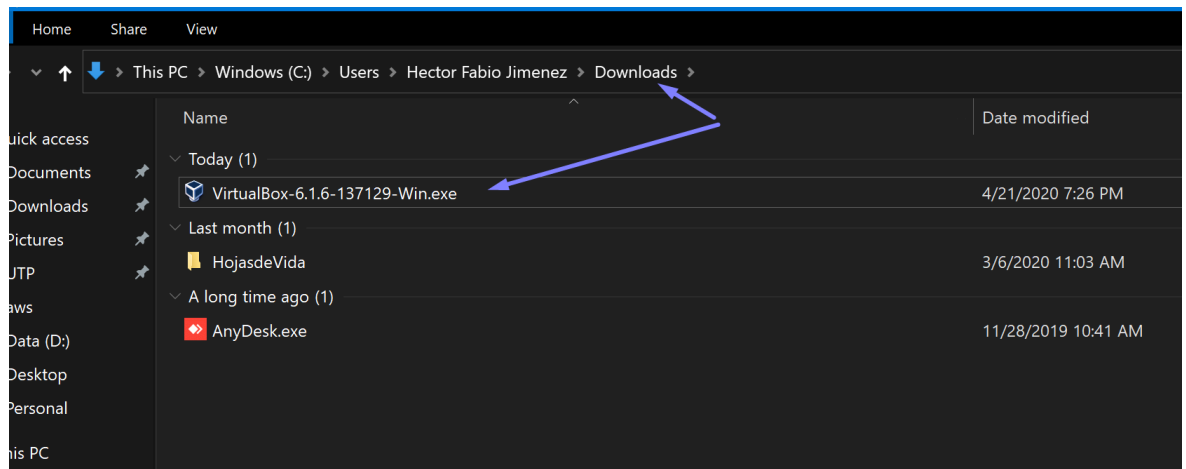


Figura 2: Instalador Virtualbox, Versión 6.1.6.

Para proceder con esto, se ejecuta el instalador como administrador y se continua con las instrucciones del setup wizard, como se muestra en la Figura.3.

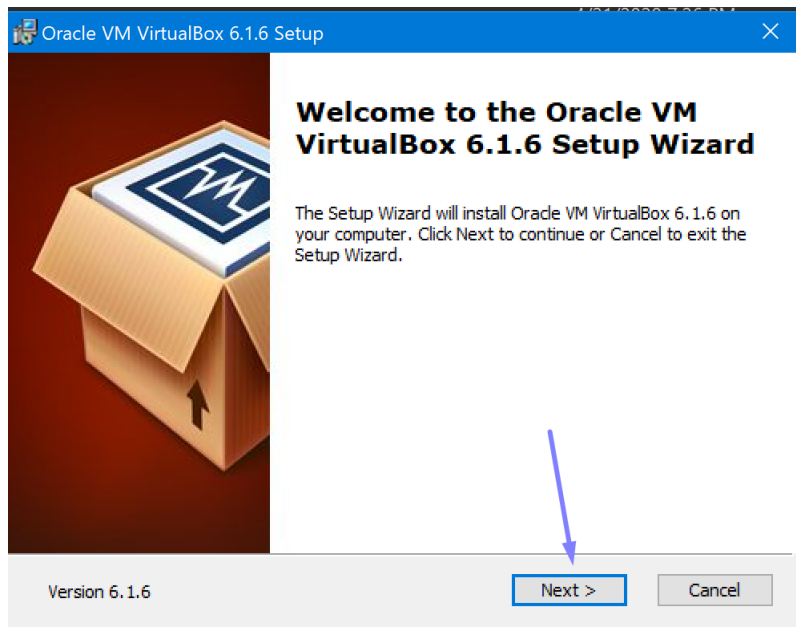


Figura 3: Ejecución de Wizard como administrador.

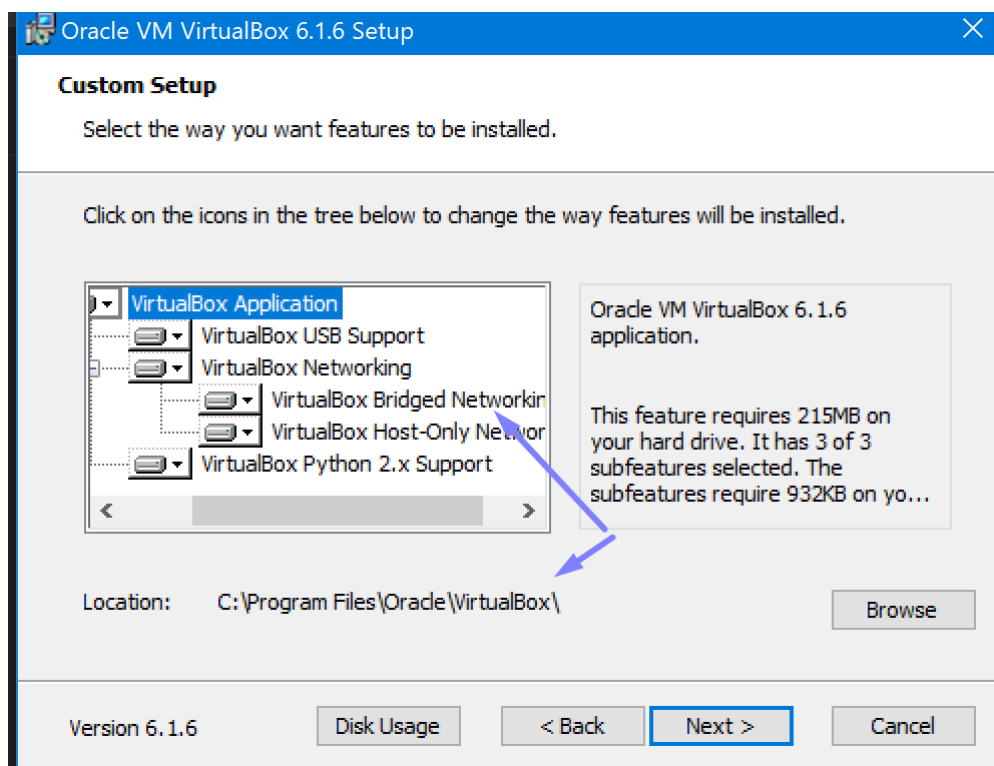


Figura 4: Es obligatorio instalar VirtualBox con las opciones resaltadas, soportando los dos modos de Interfaz de red.

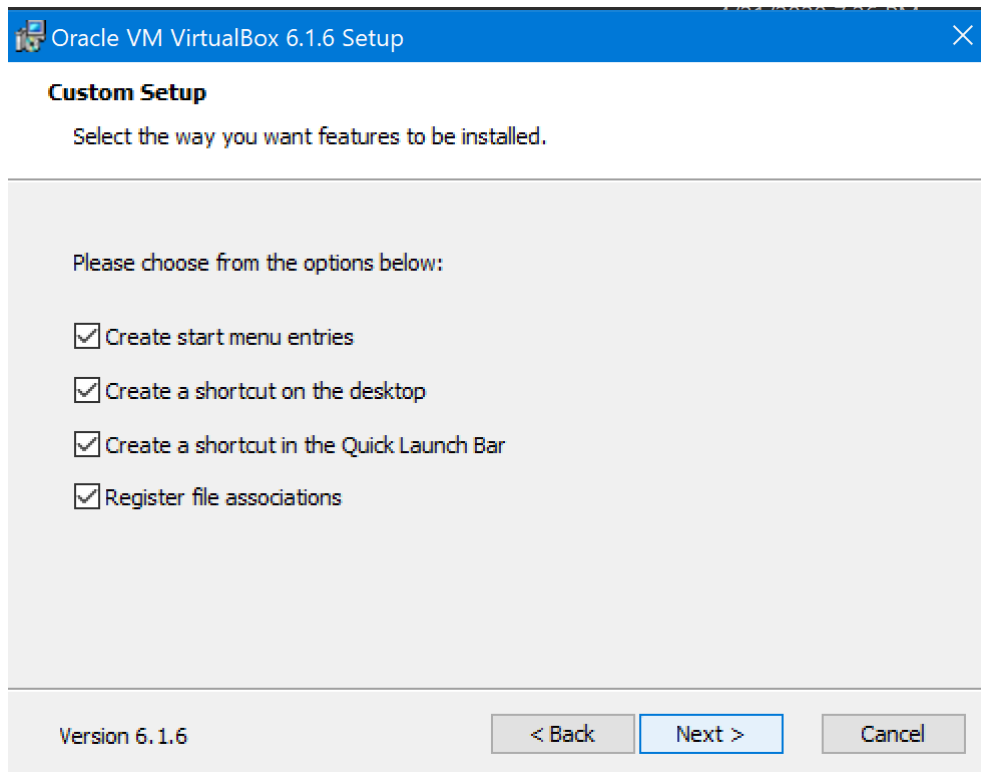


Figura 5: Se permite crear accesos directos y registro de extensiones preferencia, Wizard.

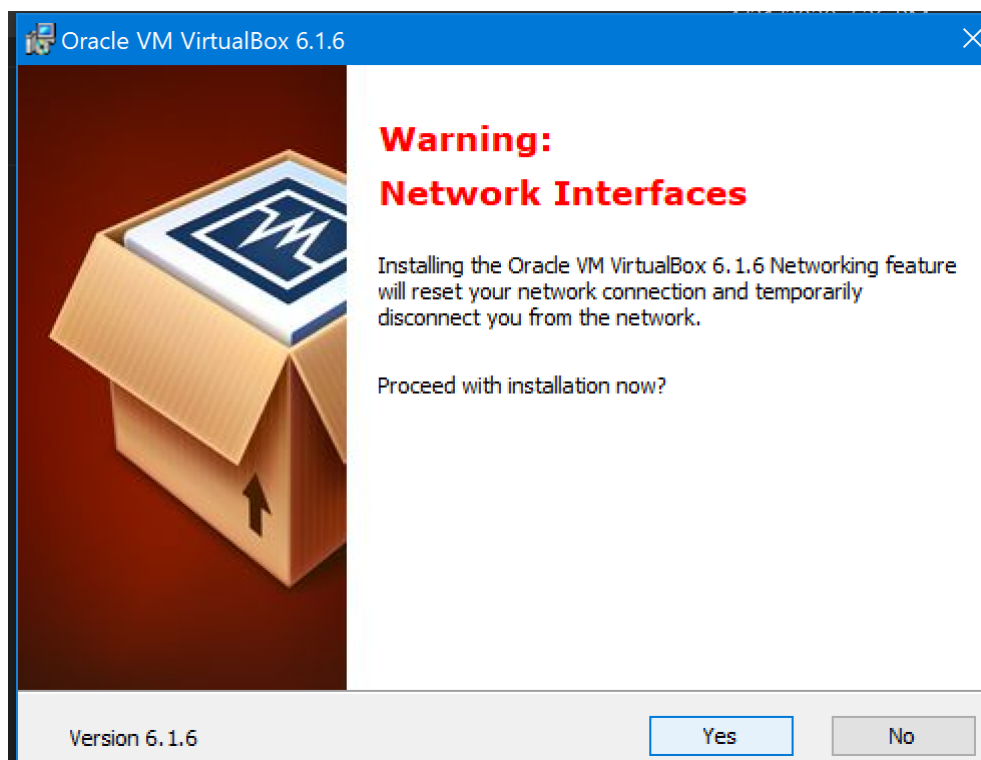


Figura 6: Características de Networking y Driver requieren reinicio de la máquina host.

De esta manera el sistema operativo ya debería contar con el hypervisor VirtualBox instalado. Una buena práctica es posteriormente ejecutar VirtualBox para validar que no existan problemas de dependencias o falta de librerías *DLL* en el sistema operativo. Se debe observar algo como en la Figura.7

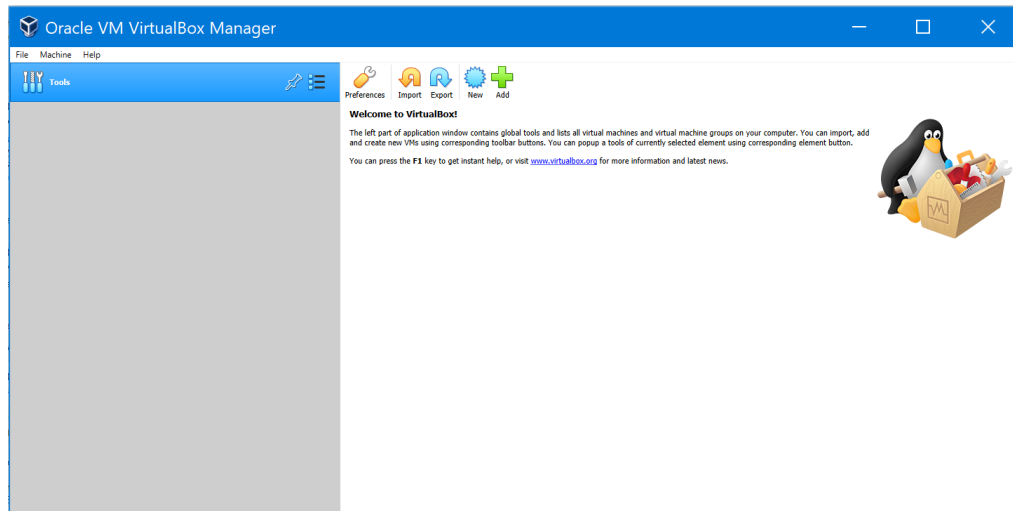


Figura 7: Instalación completa VirtualBox.

9.1.2. Instalación de Vagrant

Vagrant como se ha mencionado previamente es una herramienta de código abierto disponible para Windows, MacOS X, GNU/Linux que permite la creación y configuración de ambientes virtuales portables, repetibles y ligeros, de manera programática. Mediante scripts desarrollados en el lenguaje de programación Ruby se puede definir las características de las máquinas virtuales, entre ellas la cantidad de memoria ram, cantidad de CPUs virtuales, todas las configuraciones de red, además de tener funciones adicionales mediante algo que se conoce como "provisioners" que permite instalar software, alterar configuraciones y mucho más como parte del proceso de creación e inicialización.²

²Esto sucede al momento de realizar la ejecución del comando `vagrant up`.

Esta herramienta, utilizada para construir máquinas virtuales preconfiguradas³, fue desarrollada por Mitchell Hashimoto en el año 2010, junto a su compañero de trabajo John Bender, ambos fundarían la compañía que hoy brinda el soporte comercial **HashiCorp Inc**, actualmente con una gran variedad de productos orientados a la cultura DevOps⁴. La forma mas fácil de obtener Vagrant instalado y corriendo, según el libro oficial [17], es descargar el software del sitio web **www.vagrantup.com**

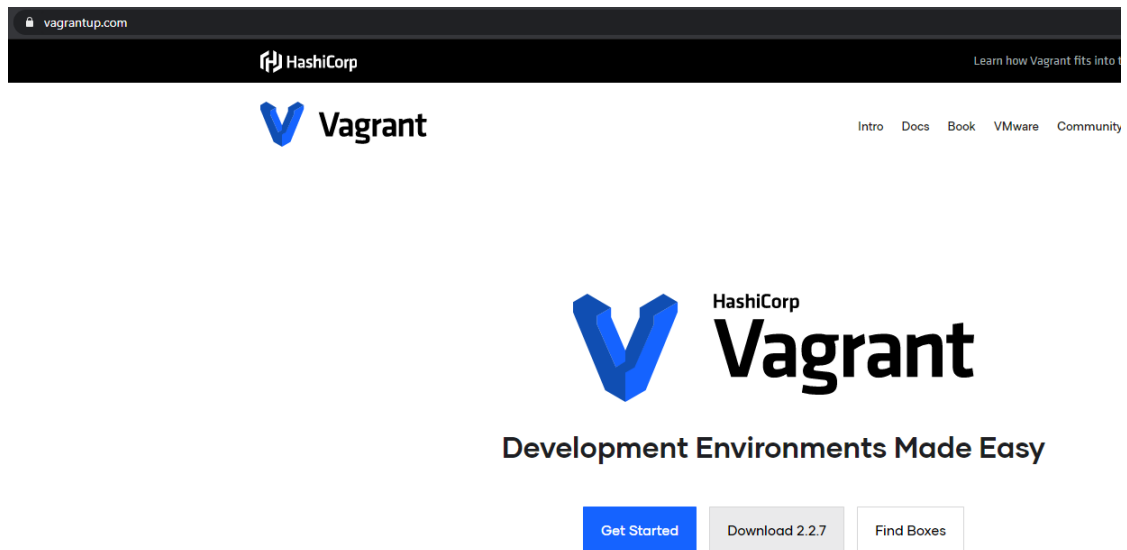


Figura 8: Sitio web oficial de vagrant, sección de descarga

Al momento de crear este documento Vagrant se encuentra en la versión 2.2.7. Durante el desarrollo de este proyecto el autor de esta tesis tuvo la posibilidad de realizar diferentes charlas en varios meetups del país:

- Manizales Tech Talks
- Bucaramanga Js
- Pereira Tech Talks

Los slides de esta presentación puede ser encontradas en el sitio web <https://slides.com/c1b3r/vagrant>.

³La preconfiguración hace referencia al uso de la imagen de un sistema operativo base, éstas pueden ser encontradas en <https://app.vagrantup.com/boxes/search>. También puede ser un término usado para referirse a la preinstalación de software.

⁴otra gran herramienta de esta compañía es packer, <https://www.packer.io/>

Es necesario, como se mencionó previamente, instalar vagrant con permisos de administrador; de la misma manera se sigue el setup wizard de Vagrant haciendo click en el boton **next**.

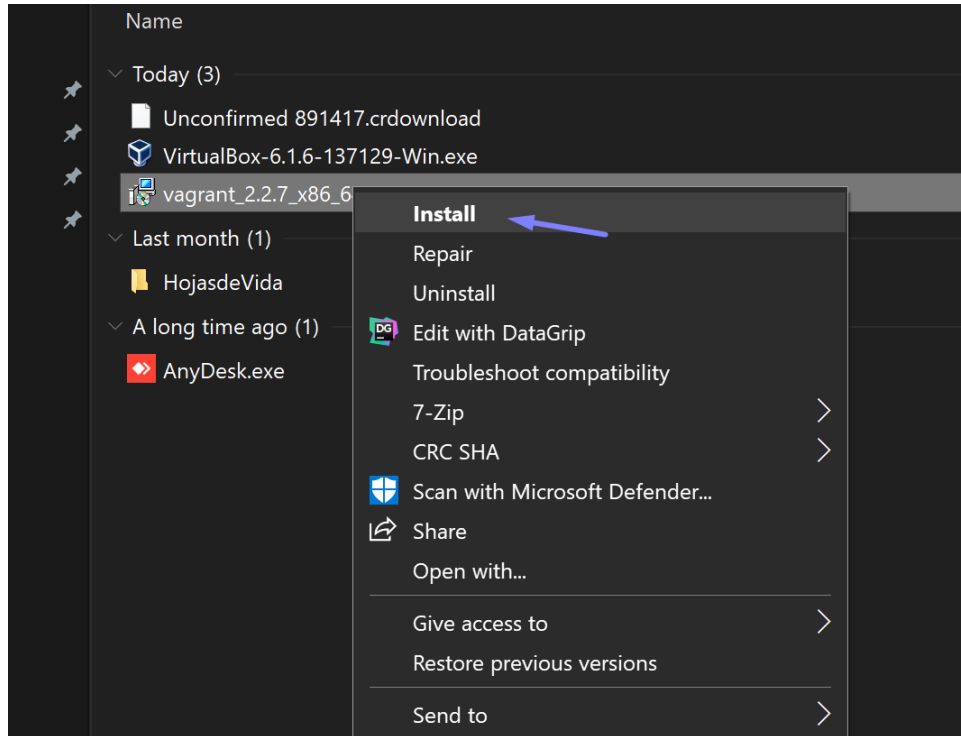


Figura 9: Instalador de Vagrant, setup wizard.

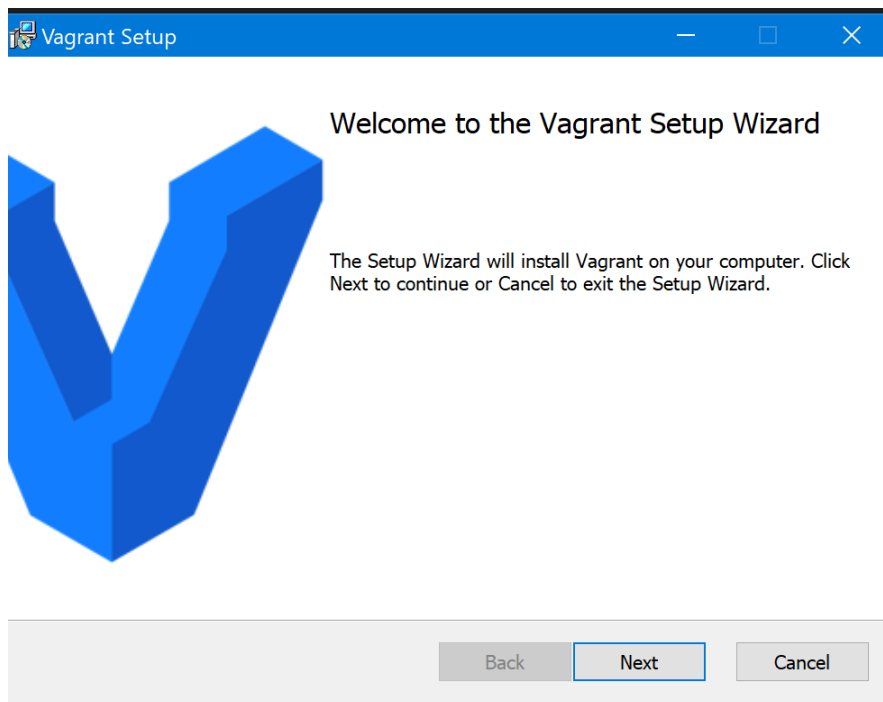


Figura 10: Setup Wizard de Vagrant.

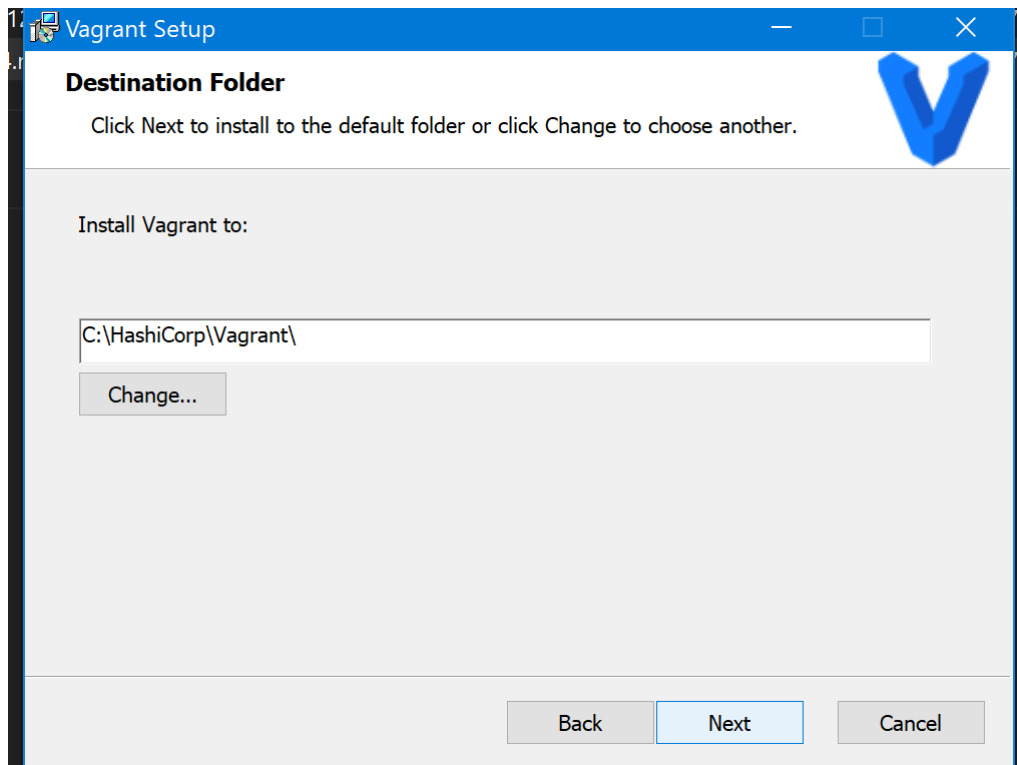


Figura 11: Selección de ubicación para instalar Vagrant. La recomendación oficial es dejar en la ruta por defecto, ya que de esta forma no habrá conflicto para definirlo dentro de las variables de entorno (la definición dentro de las variables de entorno es realizado por Vagrant de manera automática).

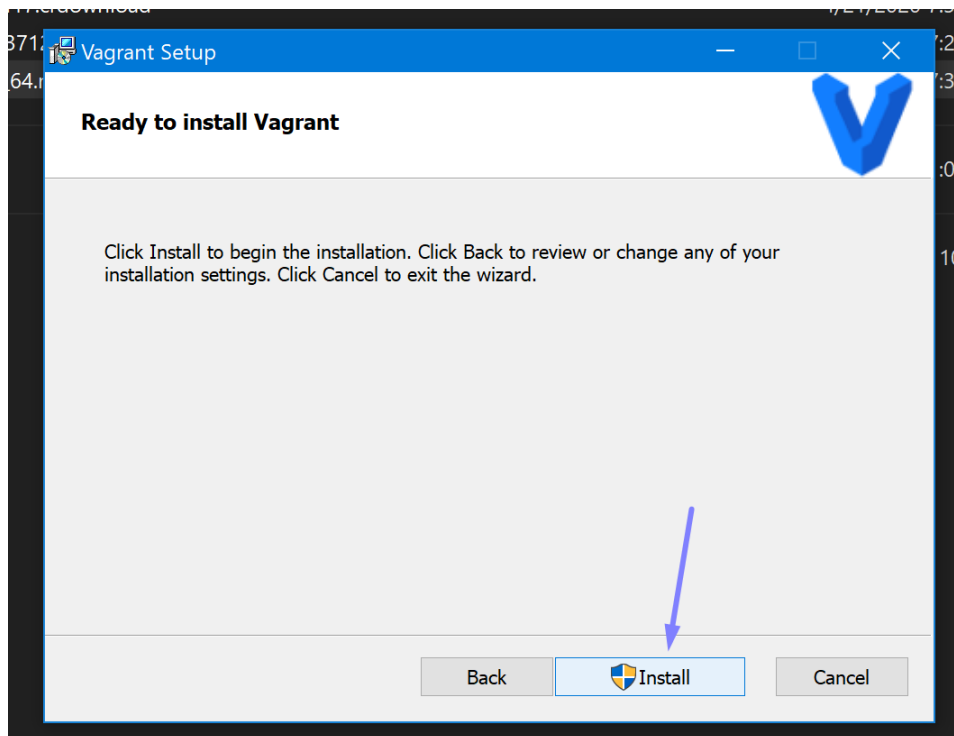


Figura 12: Paso de confirmación para el install wizard

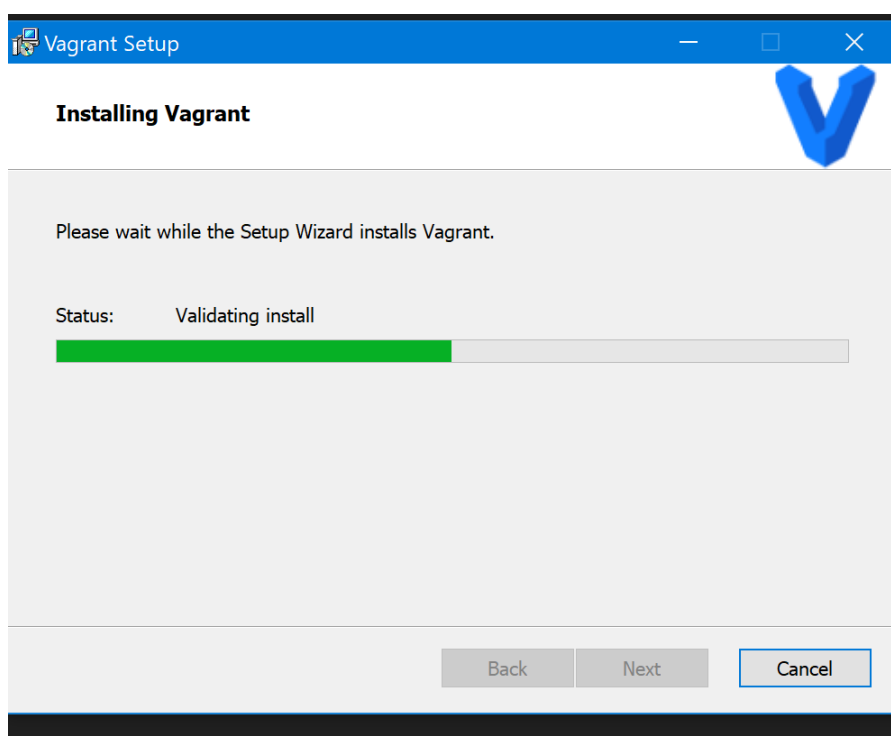
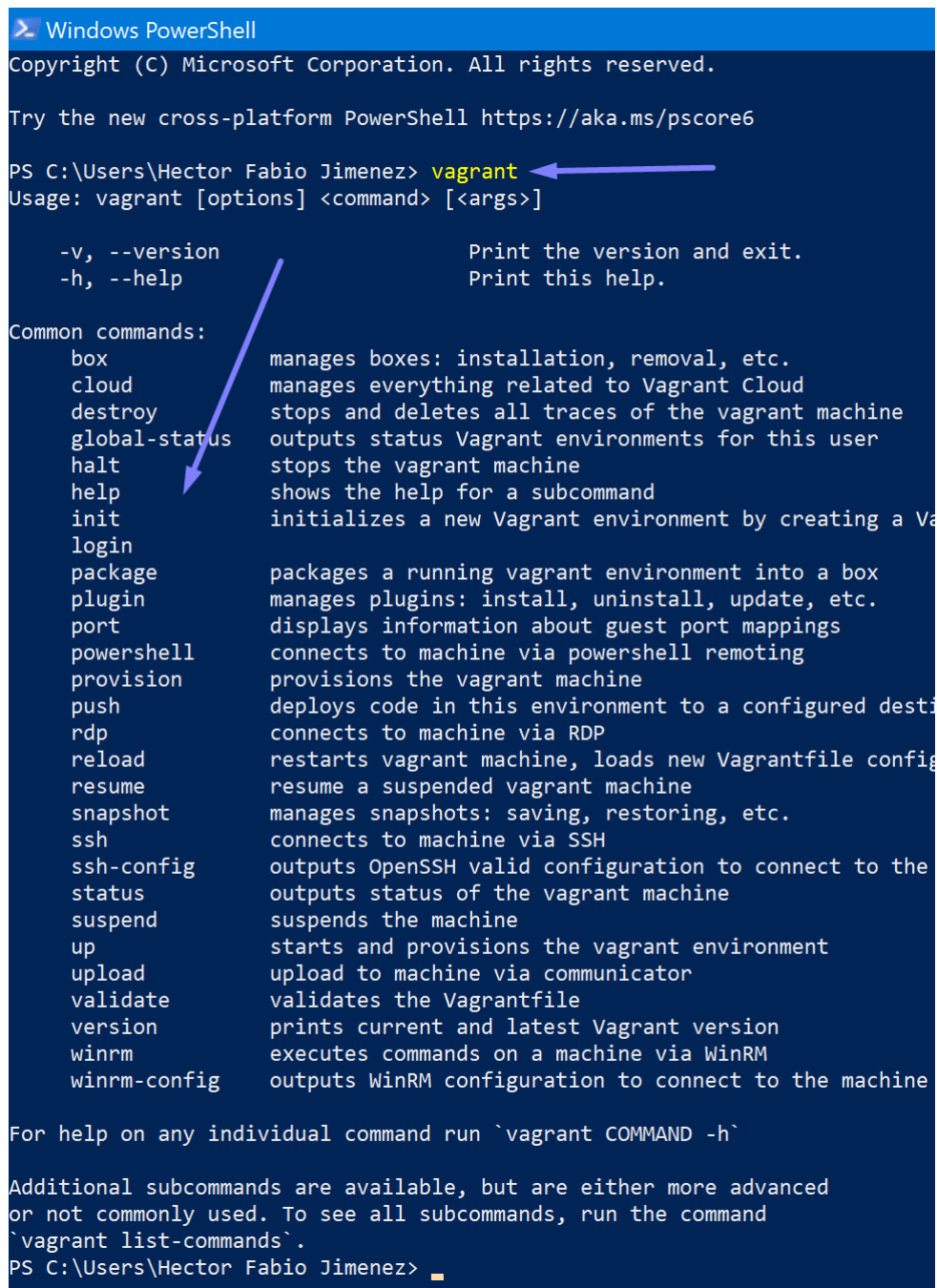


Figura 13: Instalación de Vagrant en proceso.

Al momento de que el procedimiento guiado de instalación ha terminado, se debe abrir una consola de powershell o cmd para buscar que el comando está correctamente instalado y dentro de las variables de entorno.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Hector Fabio Jimenez> vagrant
Usage: vagrant [options] <command> [<args>]

    -v, --version          Print the version and exit.
    -h, --help            Print this help.

Common commands:
    box                   manages boxes: installation, removal, etc.
    cloud                manages everything related to Vagrant Cloud
    destroy              stops and deletes all traces of the vagrant machine
    global-status        outputs status Vagrant environments for this user
    halt                stops the vagrant machine
    help                 shows the help for a subcommand
    init                initializes a new Vagrant environment by creating a Va
    login
    package              packages a running vagrant environment into a box
    plugin               manages plugins: install, uninstall, update, etc.
    port                 displays information about guest port mappings
    powershell           connects to machine via powershell remoting
    provision            provisions the vagrant machine
    push                 deploys code in this environment to a configured desti
    rdp                  connects to machine via RDP
    reload               restarts vagrant machine, loads new Vagrantfile config
    resume              resume a suspended vagrant machine
    snapshot             manages snapshots: saving, restoring, etc.
    ssh                  connects to machine via SSH
    ssh-config           outputs OpenSSH valid configuration to connect to the
    status               outputs status of the vagrant machine
    suspend              suspends the machine
    up                  starts and provisions the vagrant environment
    upload               upload to machine via communicator
    validate             validates the Vagrantfile
    version              prints current and latest Vagrant version
    winrm                executes commands on a machine via WinRM
    winrm-config         outputs WinRM configuration to connect to the machine

For help on any individual command run `vagrant COMMAND -h`

Additional subcommands are available, but are either more advanced
or not commonly used. To see all subcommands, run the command
`vagrant list-commands`.
PS C:\Users\Hector Fabio Jimenez>
```

Figura 14: Opciones disponibles de Vagrant en la línea de comandos.

9.2. Pruebas de Aprovisionamiento Vagrant Boxes

Según el sitio web de Redhat Inc [27], la etapa de aprovisionamiento se define como el conjunto de pasos requeridos para administrar el acceso a los datos y recursos, y de cómo es posible hacerlos disponibles a los usuarios y sistemas [27]. Microsoft y Vmware, por otro lado, en el contexto de las máquinas virtuales, indican que el aprovisionamiento son los diferentes mecanismos con las cuales se crean las máquinas virtuales [12] [37].

Al cumplir con los prerequisites de instalación de Vagrant y VirtualBox en el computador, es hora de proceder a correr un ejemplo muy básico donde se puede observar que el aprovisionamiento funciona adecuadamente entre Vagrant(*automatización descriptiva*) y VirtualBox(*hypervisor*).

Esta prueba consiste en lanzar una máquina virtual utilizando ambas herramientas. Para realizarlo Vagrant solo necesita tener la instancia del archivo de configuración, este es llamado (“**Vagrantfile**”), su función primaria es describir el tipo de máquina requerida por proyecto y la definición de como aprovisiona la(s) máquina(s) virtuales. Este archivo es llamado así debido a que es el nombre literal del archivo que Vagrant va a intentar leer al momento de ejecutar el comando `vagrant up`. Es necesario mencionar que Vagrant corre un solo archivo de configuración por proyecto, además sera opcional que el archivo sea versionado utilizando una herramienta como *Git*, de esta manera, un nuevo desarrollador que se integre al equipo simplemente tendrá que clonar el repositorio del proyecto y ejecutar `vagrant up` para tener el entorno de desarrollo montado y funcionando en cuestión de minutos. Por defecto, Vagrant utiliza VirtualBox como motor de máquinas virtuales o **provider**, aunque existe la opción de utilizar otros providers como VMWare Workstation (Windows) o VMWare Fusion (MacOS X) con un plugin de pago, AWS EC2 Web Services, Compute Engine de Google Cloud Platform.

La sintaxis de los archivos de Vagrant utilizan como lenguaje de programación Ruby, pero el conocimiento del lenguaje de programación no es necesario para hacer las modificaciones, dado que este trata la descripción de los recursos de una manera simple y concisa.

En los ejemplos de las Figuras Figura.16, Figura.17 se puede observar claramente la descripción de hardware.

El flujo de trabajo de vagrant está resumido en la figura Figura.15, de la que se pueden resaltar los siguientes pasos:

1. El desarrollador posee el archivo descriptivo “**Vagrantfile**” y ejecuta el comando `vagrant up`
2. Vagrant, con su analizador léxico validará las descripciones del código para Virtualbox. También revisa si es necesario ejecutar algún provisioner⁵ como un script de bash, ansible, puppet entre otros; si no, este creará todo lo necesario con el hypervisor.
3. La máquina virtual estará disponible para el desarrollo.
4. Mediante **ssh** el desarrollador tendrá acceso a todo el entorno `vagrant ssh box`

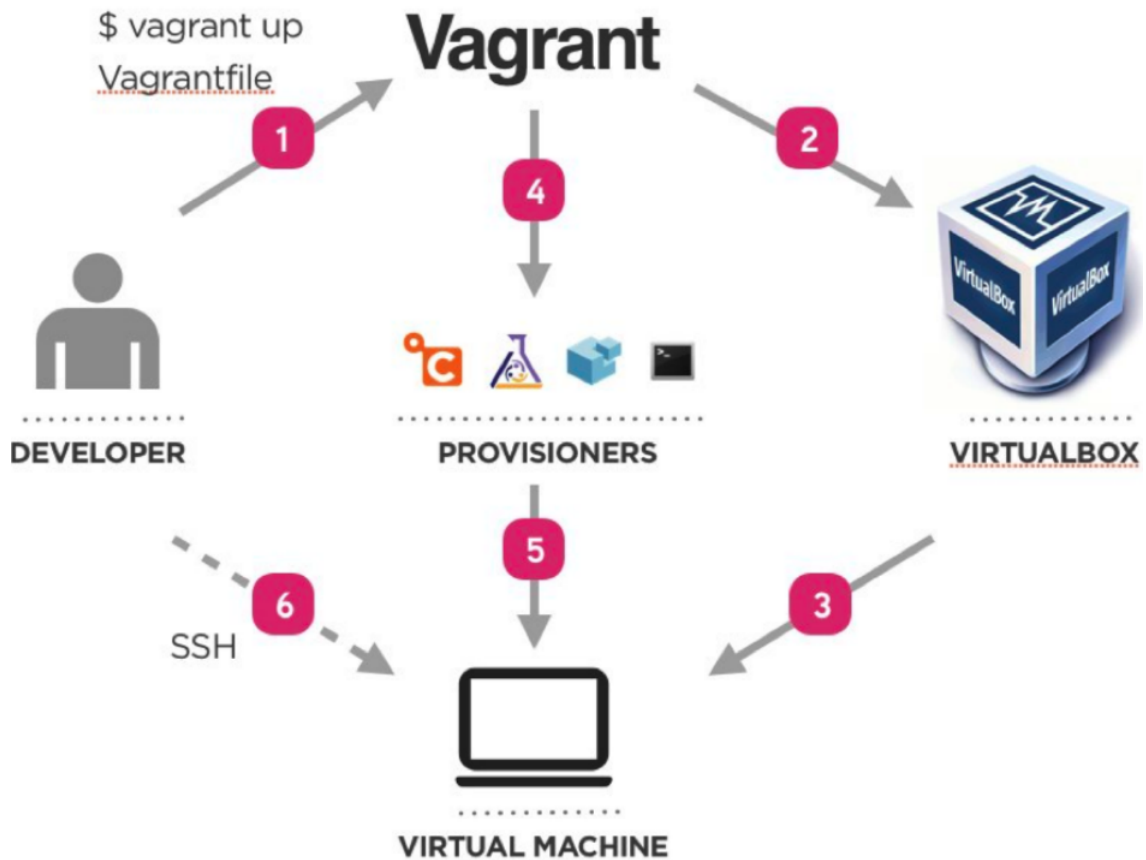


Figura 15: Flujo de Trabajo de Vagrant, Etapas.

⁵Los provisioners son opcionales, pero son de un uso común en la creación de entorno y despliegues de máquinas virtuales

```

Vagrant.configure(2) do |config|
  config.vm.box = 'bertvv/centos71'
  config.vm.hostname = 'box001'
  config.vm.network 'private_network',
    ip: 192.168.56.10

  config.vm.provision 'ansible' do |ansible|
    ansible_playbook = 'ansible/site.yml'
  end
end

```




Figura 16: Ejemplo de Vagrantfile, versión 1.

Un ejemplo de para esta prueba puede ser encontrado en Github, <https://github.com/h3ct0rjs/FichaTecnica-ProyectoGrado/blob/master/ClusterVagrant/SingleMPIMachine/Vagrantfile>

```

V Vagrantfile
1  # -*- mode: ruby -*-
2  # vi: set ft=ruby :
3  #proyecto de grado, utp
4  Vagrant.configure("2") do |config|
5    config.vm.box = "c1b3rh4ck/mpiccluster-base"
6    config.vm.box_version = "0.0.1"
7    config.vm.box_check_update = false
8    config.vm.network "public_network"
9
10   config.vm.provider "virtualbox" do |vb|
11     vb.gui = false
12     vb.memory = "1024"
13   end
14 end

```

Figura 17: Ejemplo de Vagrantfile, versión 2.

```

PS D:\Personal\UTP\VagrantLabs\Provisioning> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
=> default: Box 'c1b3rh4ck/mpiclust-base' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: 0.0.1
=> default: Loading metadata for box 'c1b3rh4ck/mpiclust-base'
    default: URL: https://vagrantcloud.com/c1b3rh4ck/mpiclust-base
=> default: Adding box 'c1b3rh4ck/mpiclust-base' (v0.0.1) for provider: virtualbox
    default: Downloading: https://vagrantcloud.com/c1b3rh4ck/boxes/mpiclust-base/versions/0.0.1/providers/virtual
ox
    default: Download redirected to host: vagrantcloud-files-production.s3.amazonaws.com
    default: Progress: 13% (Rate: 3615k/s, Estimated time remaining: 0:04:21)

```

Figura 18: resultado de importación y creación de vagrant up.

La primera vez que se realiza **vagrant up**, se realiza la validación de si la imagen base se encuentra ⁶. Si Vagrant no logra detectarla, el realiza la descarga desde un repositorio de imágenes conocido como vagrantcloud. Seguidamente realiza la configuración de los parámetros de memoria, cpu y red puestos en la maquina virtual. Para este proyecto de grado, se realizaron múltiples pruebas, por sugerencias del director de proyecto y como buena practica para minimizar los tiempos de aprovisionamiento, se realizo la creación de una imagen base que ya contiene los archivos binarios de MPI pues durante las pruebas el tiempo que tardaba el cluster en levantar todos los nodos esclavos mientras instalaba las múltiples dependencias era muy alto. ⁷

```

PS D:\Personal\UTP\VagrantLabs\Provisioning> vagrant status
Current machine states:

default                                running (virtualbox)

```

The VM is running. To stop this VM, you can run `vagrant halt` to shut it down forcefully, or you can run `vagrant suspend` to simply suspend the virtual machine. In either case, to restart it again, simply run `vagrant up`.

```

PS D:\Personal\UTP\VagrantLabs\Provisioning>

```

Figura 19: Aprovisionamiento Exitoso, Estado de la máquina virtual.

Cuando todo el proceso de aprovisionamiento termina, podremos comprobar el

⁶ver linea de código 5 de la Figura 17

⁷La imagen base puede ser encontrada en <https://app.vagrantup.com/c1b3rh4ck/boxes/mpiclust-base>.

estado utilizando `vagrant status`, en el cual se debe observar lo que se presenta en la Figura 19 con status de *running*.

```
PS D:\Personal\UTP\VagrantLabs\Provisioning> vagrant status
Current machine states:

default              running (virtualbox)

The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down forcefully, or you can run `vagrant suspend` to simply
suspend the virtual machine. In either case, to restart it again,
simply run `vagrant up`.
PS D:\Personal\UTP\VagrantLabs\Provisioning> vagrant ssh
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-179-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * "If you've been waiting for the perfect Kubernetes dev solution for
  macOS, the wait is over. Learn how to install Microk8s on macOS."

  https://www.techrepublic.com/article/how-to-install-microk8s-on-macos/

0 packages can be updated.
0 updates are security updates.

New release '18.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon May 18 17:11:06 2020 from 10.0.2.2
vagrant@master-cluster:~$
```

Figura 20: Conexión exitosa a la máquina virtual aprovisionada, `vagrant ssh`.

Finalmente la conexión a la máquina virtual se ejecuta lanzando el comando `vagrant ssh`.

9.3. Pruebas de Networking y Direcccionamiento

Oracle VM VirtualBox provee hasta 8 tarjetas virtuales Ethernet PCI para cada máquina virtual. Cada tarjeta, se puede seleccionar individualmente lo siguiente:

- Hardware que será virtualizado.

- Modo de Virtualización que la tarjeta usará en su modo de operación con respecto al hardware de red en la máquina host entre los modos Nat, Bridge, Red Interna.

Siempre en la interfaz gráfica de VirtualBox se podrá configurar hasta 4 tarjetas tal cual como se muestra en la Figura.21, las otras 4 tarjetas de red pueden ser configuradas utilizando las opciones de línea de comandos provista por VirtualBox *VBoxManage modifyvm* [13].

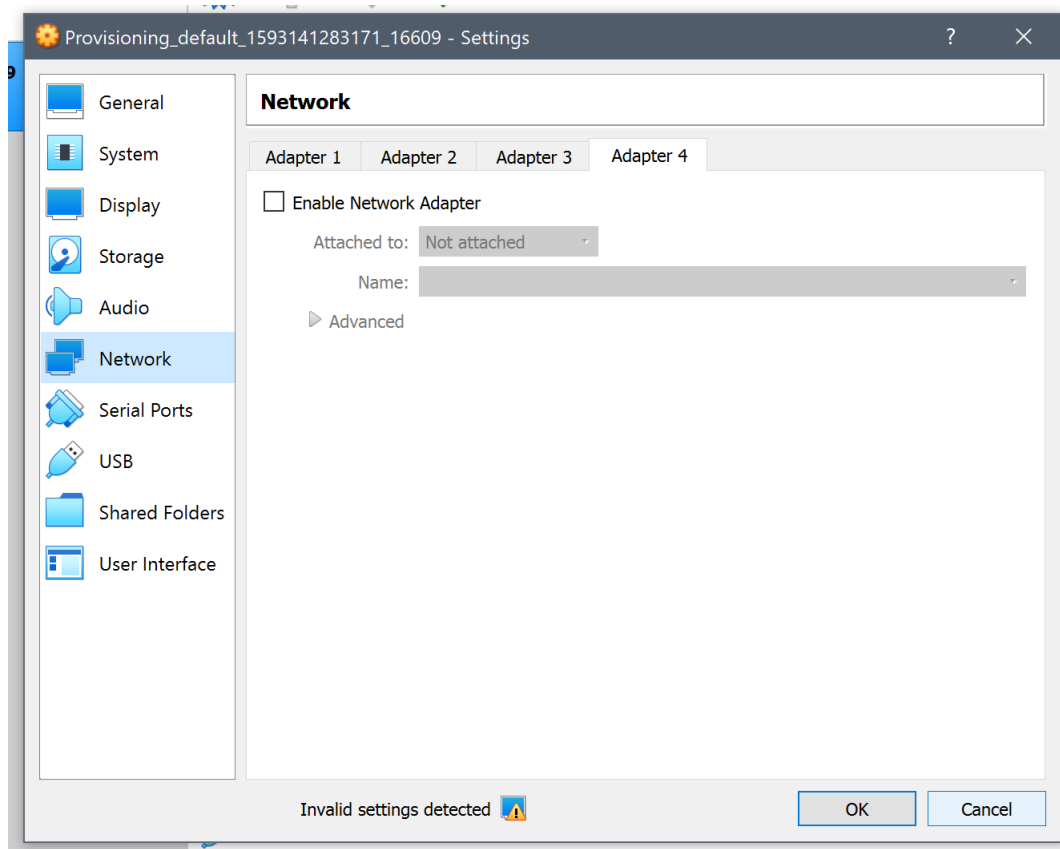


Figura 21: Sección de red, máquina virtual aprovisionada.

Como se mencionó previamente, los modos de virtualización de las tarjetas de red de VirtualBox definen la visibilidad en un segmento de red y el alcance de las máquinas virtuales, también las capacidades que tendrá la máquina virtual a nivel de networking. A continuación se presenta un resumen de los modos, donde se denota si las máquinas virtuales alcanzan el equipo host, o a varias máquinas virtuales entre sí o con una red específica.

Table 6.1. Overview of Networking Modes

Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	-	-
Internal	-	-	+	-	-
Bridged	+	+	+	+	+
NAT	+	Port forward	-	+	Port forward
NATservice	+	Port forward	+	+	Port forward

Figura 22: Conexión a la máquina virtual aprovisionada, `vagrant ssh`.

Por ejemplo, si se mira el modo Host-Only de la Figura.22 se puede observar:

- La conexión entre la máquina virtual y el Sistema operativo se puede establecer en ambos sentidos, según las flechas indicativas.
- La conexión entre múltiples máquinas virtuales se puede realizar en ambos sentidos.
- La conexión con redes que creamos en Virtualbox no se puede alcanzar.

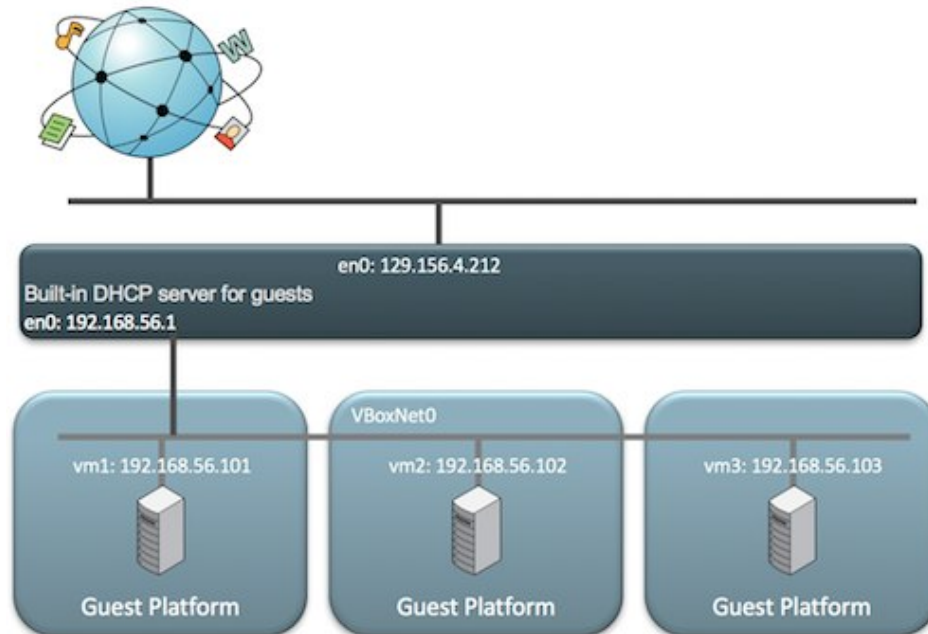


Figura 23: Conexión usando el modo Host Only, Implementación del Cluster MPI, cada máquina virtual Guest queda en el mismo segmento de red permitiendo comunicación bidireccional.

Además de los diferentes modos provistos por VirtualBox, este también provee un administrador de redes, donde se permite administrar los segmentos y las redes creadas virtualmente. En la Figura.24 se observa que cada vez que se crea una red LAN utilizando VirtualBox Network Manager este crea a su vez los dispositivos de red virtuales en el sistema operativo host. Si estos dispositivos no se encuentran creados al momento de aprovisionar el clúster, Vagrant realiza su creación de manera automática.

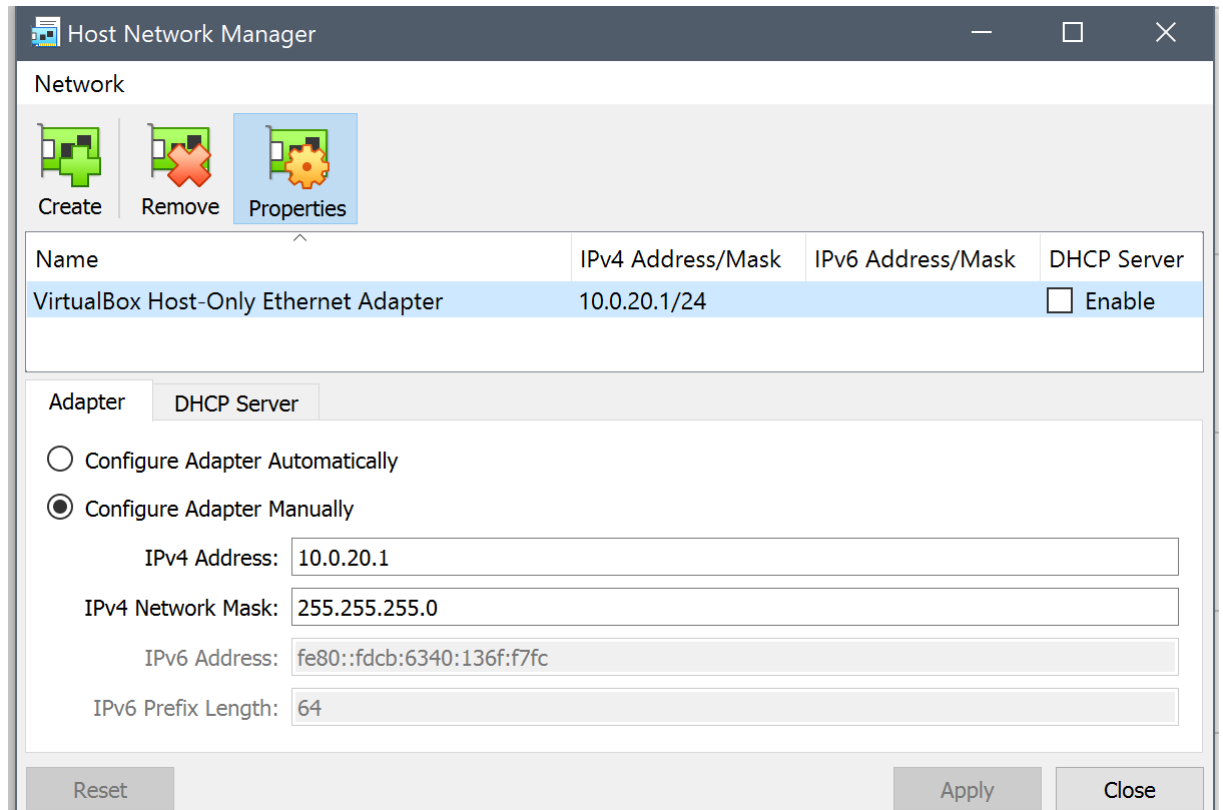


Figura 24: Conexión a la máquina virtual aprovisionada, `vagrant ssh`.

9.3.1. Opciones de Configuración Programáticas Vagrant

Vagrant a través de su declaración programática expone opciones de alto nivel para manipular las configuraciones de las tarjetas de red y las redes creadas por VirtualBox, evitando tener que realizar la configuración manualmente en la interfaz gráfica de Virtualbox, entre ellas:

- Configuraciones de PortForwarding.
- Configuraciones de múltiples segmentos de red por tarjeta virtual.

- Creación de subredes privadas.

```
Vagrant.configure("2") do |config|  
  # ...  
  config.vm.network "forwarded_port", guest: 80, host: 8080  
end
```

Figura 25: Port Forwarding .

En la Figura.25 Se puede observar como se realiza de manera declarativa la exposición del puerto **80** en la máquina virtual y que se reflejará en el puerto **8080** en la máquina host.

También declarativamente se puede establecer diferentes puertos, además del tipo de protocolo como se observa en la Figura.28

```
Vagrant.configure("2") do |config|  
  config.vm.network "forwarded_port", guest: 2003, host: 12003, protocol: "tcp"  
  config.vm.network "forwarded_port", guest: 2003, host: 12003, protocol: "udp"  
end
```

Figura 26: Multiple Port Forwarding, con puertos diferentes en máquina host e invitado, especificando protocolo.

Vagrant también permite especificar los tipos de redes, como por ejemplo las redes privadas que le permiten acceder a su máquina virtual por alguna dirección que no sea públicamente accesible desde Internet o en la misma LAN de la máquina host. En general, esto significa que la máquina virtual obtiene una dirección en el espacio de direcciones privadas.

```
Vagrant.configure("2") do |config|  
  config.vm.network "private_network", type: "dhcp"  
end
```

Figura 27: Declaración de red privada, asignación de IP por dhcp.


```
Vagrant.configure("2") do |config|
  config.vm.network "private_network", ip: "192.168.50.4"
end
```

Figura 28: Declaración de red privada, asignación de dirección ipv4 estática.

Una configuración de estas pruebas de networking puede ser encontrada en Github, <https://github.com/h3ct0rjs/FichaTecnica-ProyectoGrado/blob/master/ClusterVagrant/NetworkingLab/Vagrantfile>.

9.4. Aprovisionamiento de Múltiples Maquinas Virtuales

Vagrant además de las opciones anteriores también permite realizar la instanciación de la cantidad que se desee de máquinas virtuales en un mismo Vagrantfile. Nótese cómo en la Figura.29 la definición quiere crear dos máquinas virtuales al mismo tiempo: una con el nombre de **web** y otra con el nombre **db**, la primera basada en una imagen de *apache* y la segunda de *mysql*.

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell", inline: "echo Hello"

  config.vm.define "web" do |web|
    web.vm.box = "apache"
  end

  config.vm.define "db" do |db|
    db.vm.box = "mysql"
  end
end
```

Figura 29: Defunción de Múltiples máquinas virtuales, `vagrant ssh`.

Por otra parte, en la definición de las máquinas virtuales la herramienta permite realizar iteraciones como se muestra en la Figura.30, donde se crean tres máquinas virtuales y se da el nombre de *node-#i*, De esta misma manera se realiza la configuración de los nodos del cluster MPI.

```
(1..3).each do |i|
  config.vm.define "node-#{i}" do |node|
    node.vm.provision "shell",
      inline: "echo hello from node #{i}"
  end
end
```

Figura 30: Conexión a la máquina virtual aprovisionada, `vagrant ssh`.

Una configuración de estas pruebas de aprovisionamiento múltiple puede ser encontrada en Github,

- <https://github.com/h3ct0rjs/FichaTecnica-ProyectoGrado/blob/master/ClusterVagrant/MultipleNodes/SimpleMultipleNodes/VagrantFile>.
- <https://github.com/h3ct0rjs/FichaTecnica-ProyectoGrado/blob/master/ClusterVagrant/MultipleNodes/VarMultipleNodes/VagrantFile>.

9.5. Sistema de Archivos Compartido clúster MPI

El protocolo NFS ha existido durante casi mas de 20 años desde su primer RFC publico *rfc1094* [25], es el protocolo de intercambio de archivos de red de facto para sistemas operativos Unix y Gnu/Linux. El protocolo NFS se encuentra en su cuarta versión y esta a cargo de investigadores de todo el mundo, con desarrolladores y científicos de todo el mundo. El protocolo NFS proporciona acceso remoto y transparente a archivos y recursos compartidos en red, este está diseñado para ser portable y multiplataforma además de ser soportado en diferentes tipos de máquinas, sistemas operativos, arquitecturas de redes y protocolos de transporte. Su portabilidad y funcionamiento se logra mediante el uso de primitivas RPC integradas junto con la representación de datos externos (XDR, eXternal Data Representation).

En la actualidad la versión estable de este protocolo es la **NFSv4**, con importantes mejoras como el soporte para manejar sistemas de archivos de 64bits, escritura de archivos asíncronas, funcionamiento a través de firewalls y soporte para configuraciones de listas de control de acceso **NFSv4**. El clúster MPI requiere de

un sistema de archivos compartido entre el nodo maestro y los nodos esclavos para poder compartir el binario que se ejecutará como parte de las llamadas MPI.

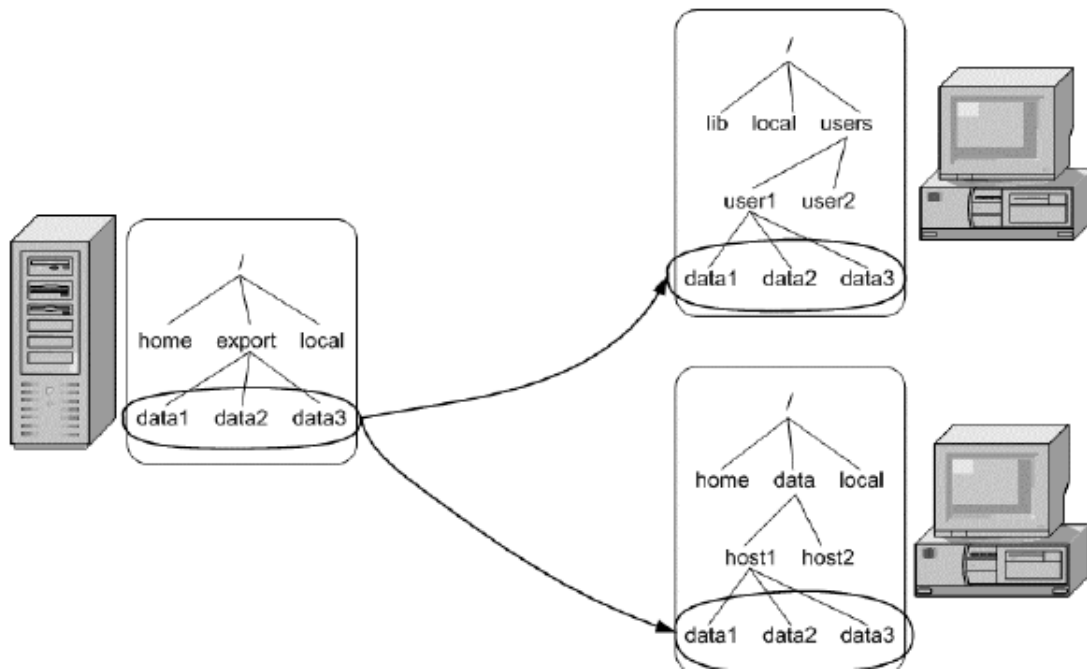


Figura 31: Funcionamiento de NFS.

El protocolo NFS trabaja como un servicio sobre un sistema de archivos real, trabaja con archivos en vez de trabajar con bloques, a manera de un sistema de archivos distribuido. En la Figura.31 se observa claramente como en esta arquitectura cliente servidor, el servidor equipo de la izquierda exporta un directorio de nombre `/export`, debajo de este hay varios archivos (`data1`, `data2`, `data3`). Al lado derecho se puede observar dos equipos que cumplen la función de cliente, estos mediante un cliente nfs montan el recurso compartido por el servidor; cuando el cliente termina de negociar los diferentes parámetros permite que los clientes tengan acceso a los mismos datos del servidor. Notese que cada cliente puede elegir la ubicación en la que desea montar el sistema de archivos, en el caso del cliente de la parte superior su punto de montaje es `/users/user1/` y el otro `/data/host1/`.

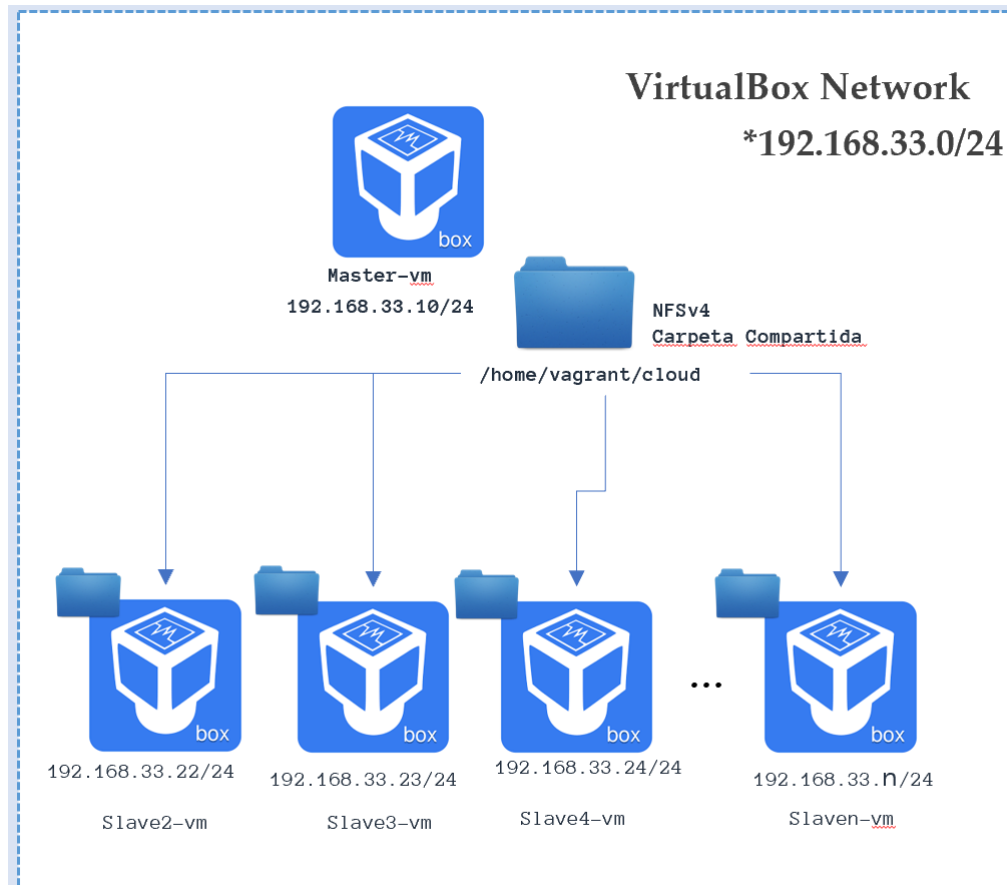


Figura 32: Funcionamiento de NFS, ejemplo basico NFSv3.

Para el desarrollo de este trabajo, como se menciona previamente es necesario realizar el montaje de un sistema de archivos compartido, este permitirá que almacenemos los binarios de los programas con las sentencias de MPI. Se ha elegido la ubicación */home/vagrant/cloud* para la instalación del sistema de archivos compartido, esto se hace en un sistema operativo Ubuntu 16.04, tal cual como se muestra en los comandos de la Figura.33

```
echo -e " Installing NFS on Master Node"
mkdir /home/vagrant/cloud #Create the cloud folder to shared accross the cluster :)
chown vagrant:vagrant /home/vagrant/cloud
apt-get update
apt install -y nfs-server
echo "/home/vagrant/cloud *(rw,sync,no_root_squash,no_subtree_check)">>/etc/exports
exportfs -a
service nfs-kernel-server restart
df -h
```

Figura 33: Instalación Servidor NFSv4.

Se observa como en la máquina que actúa como servidor se instala el paquete **nfs-server** al cual se le pasan todas las opciones para el recurso compartido, en este caso :

- **rw** : Permisos para lectura y escritura en el folder.
- **sync** : Permisos para lectura y escritura en el folder.
- **no_root_squash** : No se desea que el usuario root del equipo que monta el sistema de ficheros equivalga al root del sistema que lo exporta.
- **no_subtree_check** : Mejora el performance, deshabilitando la revisión del árbol del sistema de archivos en todo los clientes y servidor.

```
echo "/home/vagrant/cloud * (rw, sync, no_root_squash, no_subtree_check)" >>
/etc/exports
```

En todos los clientes, se realiza la instalación del paquete **nfs-common** con el fin de poder montar el sistema de archivo exportado por el servidor.

```
echo -e "Installing NFS on Worker Node"
apt update
apt install -y nfs-common
mkdir /home/vagrant/cloud
chown vagrant:vagrant /home/vagrant/cloud
mount -t nfs master-vm:/home/vagrant/cloud /home/vagrant/cloud
echo "master-vm:/home/vagrant/cloud /home/vagrant/cloud nfs defaults 0 0">>/etc/fstab
df -h
```

Figura 34: Configuración NFSv4, Clientes del Cluster.

Para los clientes basta con agregar una línea donde se indica al cliente que monte todos los archivos compartidos por el servidor en la carpeta local con las configuraciones por defecto.

9.6. Configuraciones de Seguridad

Las configuraciones de seguridad realizadas para este proyecto son técnicas y métodos ampliamente difundidos por los desarrolladores y administradores de las distribuciones de sistemas operativos GNU/Linux [19] [33] [26], entre ellas las siguientes:

- Documentar la información de las máquinas virtuales en el clúster, recolectando datos como el nombre de la máquina, dirección IP, dirección MAC y el número de asset (recurso).
- Al tratarse de máquinas virtuales, es necesario asegurar que el sistema de arranque no sea modificado por personas externas al hypervisor.
- Las máquinas virtuales deben contar con las últimas actualizaciones que el sistema operativo tenga disponibles, esto se logra mediante la ejecución, en el caso de la distribución Ubuntu, del comando *apt-get update*.
- Deshabilitar servicios innecesarios del sistema, que vienen por defecto.

Realizar un listado de usuarios permitidos para la ejecución de programas MPI en cada uno de los nodos del clúster y se configuran las llaves de seguridad comunes para alcanzar el nodo maestro.

9.7. Pruebas de cl ster MPI Multimaquina

Finalmente despu s de haber realizado la comprensi n total de los diferentes elementos que se configuraron a continuaci n en la Figura.35 se muestra una arquitectura de nuestro cluster MPI Multimaquina y Multinucleo.

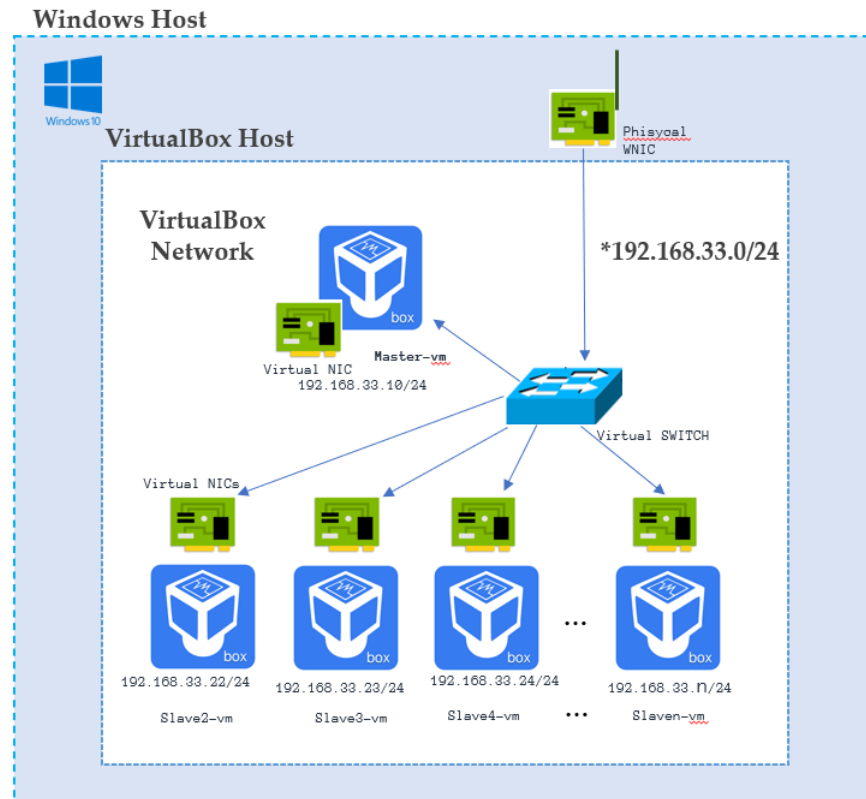


Figura 35: Arquitectura funcional cluster MPI, virtual switch .

Como se observa en la Figura.35 tenemos una arquitectura compuesta por un nodo maestro de nombre **Master-vm** con direcci n ipv4 192.168.33.10 perteneciente al segmento de red 192.168.33.0/24 en esta maquina virtual se instala el servidor de archivos NFSv4 (ver Figura.33, Figura.32), para las pruebas realizadas a este nodo de ahora en adelante se le asignaron 2 cores de CPU, 1GB de memoria ram y tambi n un segmento de red privada. De manera program tica es necesario que el nodo maestro tenga acceso a todos los nodos **Slavei-vm** que ser n de apoyo prestando sus recursos en la soluci n a tareas que se corran en el cluster. Para lograr esto fue necesario implementar un cliente y servidor *ssh* disponible con el gestor de paquetes del sistema operativo *apt-get* el paquete mas conocido *OpenSSH-Server* configurando un par de llaves comunes a todos los nodos, en las pruebas realizadas

fue necesario al momento de aprovisionar los nodos deshabilitar una opción de las maquinas que se encarga de validar en la primera conexión si existe fingerprint disponible, además esto nos evita que durante el aprovisionamiento nos genere un error de conexión.

```
echo "#SSH Keys Configuration"

if [[ ! -e /vagrant/id_rsa ]]; then
  ssh-keygen -t rsa -b 4096 -f /vagrant/id_rsa -N ""
fi
(echo; cat /vagrant/id_rsa.pub) >> /home/vagrant/.ssh/authorized_keys
cp /vagrant/id_rsa* /home/vagrant/.ssh/
chown vagrant:vagrant /home/vagrant/.ssh/id_rsa*
chmod -R 600 /home/vagrant/.ssh/authorized_keys
chmod 400 /home/vagrant/.ssh/id_rsa
touch /home/vagrant/.ssh/config
echo 'Host 192.168.33.*' >> /home/vagrant/.ssh/config
echo 'StrictHostKeyChecking no' >> /home/vagrant/.ssh/config
echo 'UserKnownHostsFile /dev/null' >> /home/vagrant/.ssh/config
echo 'StrictHostKeyChecking no' >> /etc/ssh/ssh_config
chown vagrant:vagrant /home/vagrant/.ssh/config
```

Figura 36: StrictHostChecking, Conexiones SSH deshabilitado .

También durante los experimentos realizados, se nota que en algunos momentos el cluster MPI no funcionaba debido a la no instalación correcta del cliente NFS y servidor. Se realizo una revisión detalla de los diferentes problemas encontrados en la salida de Vagrant, observando que el sistema operativo Ubuntu la primera vez que inicializa utiliza un proceso denominado cloud-init el cual a su vez llame una primera actualización programada de paquetes. Al estar esta tarea en ejecución se presenta un interbloqueo el cual no nos permite realizar la instalación con el comando apt.

```
apt install nfs
root@master-vm:/home/vagrant# ps aux|grep apt
root    1294  0.0  0.0  4500   752 ?        Ss   03:03   0:00 /bin/sh /usr/lib/apt/apt.systemd.daily update
root    1299  0.0  0.1  4500  1692 ?        S    03:03   0:00 /bin/sh /usr/lib/apt/apt.systemd.daily lock_is_held update
._apt   2538  0.0  0.6  56616  6164 ?        S    03:04   0:00 /usr/lib/apt/methods/http
._apt   2539  0.2  0.6  56620  6260 ?        S    03:04   0:00 /usr/lib/apt/methods/http
root    3040  0.0  0.0  12940   924 pts/0    St+  03:09   0:00 grep --color=auto apt
root@master-vm:/home/vagrant# apt install -y nfs-server
E: Could not get lock /var/lib/dpkg/lock-frontent - open (11: Resource temporarily unavailable)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontent), is another process using it?
```

Figura 37: Interbloqueo de procesos, Fallo de Instalación de Servicio NFS .

En la consulta realizada, se determina que la mejor manera de resolver este inconveniente es deshabilitar el servicio de **systemd apt-daily.service** al momento de lanzar la ejecución y espera que no exista ningún otro proceso haciendo uso de los archivos protegidos usados por apt.


```

systemctl stop apt-daily.service
systemctl kill --kill-who=all apt-daily.service

# wait until `apt-get updated` has been killed
while ! (systemctl list-units --all apt-daily.service | egrep -q '(dead|failed)')
do
|   sleep 0.5;
done

```

Figura 38: Solución Programática. Se Deshabilita Servicio problemático.

Para poner en marcha el cluster es necesario realizar la ejecución de los comandos que se muestran en la siguiente figura.

```

Type 'help' to get help.

PS C:\Users\h3ct0rjs> cd D:\PERSONAL\UTP\FichaTecnica-ProyectoGrado\ClusterVagrant\ClusterMPICH\
PS D:\PERSONAL\UTP\FichaTecnica-ProyectoGrado\ClusterVagrant\ClusterMPICH> ls

Directory: D:\PERSONAL\UTP\FichaTecnica-ProyectoGrado\ClusterVagrant\ClusterMPICH

Mode                LastWriteTime         Length Name
----                -
d-----          15/09/2020 11:06 p. m.             Version1
d-----          16/09/2020 11:57 a. m.             Version2
-a----          10/03/2020 10:11 a. m.             227 readme.md

PS D:\PERSONAL\UTP\FichaTecnica-ProyectoGrado\ClusterVagrant\ClusterMPICH> cd .\Version1\
PS D:\PERSONAL\UTP\FichaTecnica-ProyectoGrado\ClusterVagrant\ClusterMPICH\Version1> vagrant status
Current machine states:

master-vm           not created (virtualbox)
slave2-vm           not created (virtualbox)
slave3-vm           not created (virtualbox)
slave4-vm           not created (virtualbox)
slave5-vm           not created (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
PS D:\PERSONAL\UTP\FichaTecnica-ProyectoGrado\ClusterVagrant\ClusterMPICH\Version1> vagrant up

```

Figura 39: Lanzamiento de Cluster estado de los nodos no creados.

```

PS D:\PERSONAL\UTP\FichaTecnica-ProyectoGrado\ClusterVagrant\ClusterMPICH\Version1> vagrant status
Current machine states:

master-vm           running (virtualbox)
slave2-vm           running (virtualbox)
slave3-vm           running (virtualbox)
slave4-vm           running (virtualbox)
slave5-vm           running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
PS D:\PERSONAL\UTP\FichaTecnica-ProyectoGrado\ClusterVagrant\ClusterMPICH\Version1> vagrant ssh master-vm

```

Figura 40: Lanzamiento de Cluster en Marcha. Todos los nodos se encuentran en ejecución.

Finalmente se realiza la conexión al nodo master-vm y en este nos dirigimos a la carpeta nfs para este caso con nombre **cloud** compartida con todos los nodos.

```
vagrant@master-vm:~$ cd cloud/
vagrant@master-vm:~/cloud$ mpicc -o mpi_hello mpi_hello.c
vagrant@master-vm:~/cloud$ ls
machinefile  mpi_hello  mpi_hello.c
vagrant@master-vm:~/cloud$ ./mpi_hello
Hello world from processor master-vm, rank 0 out of 1 processors
vagrant@master-vm:~/cloud$ mpirun -np 6 ./mpi_hello
Hello world from processor master-vm, rank 1 out of 6 processors
Hello world from processor master-vm, rank 4 out of 6 processors
Hello world from processor master-vm, rank 0 out of 6 processors
Hello world from processor master-vm, rank 5 out of 6 processors
Hello world from processor master-vm, rank 3 out of 6 processors
Hello world from processor master-vm, rank 2 out of 6 processors
vagrant@master-vm:~/cloud$
```

Figura 41: Lanzamiento de Cluster en Marcha. Prueba básica de conectividad máquina master-vm.

```
vagrant@master-vm:~/cloud$ cat machinefile
master-vm
slave2-vm
slave3-vm
slave4-vm
slave5-vm
vagrant@master-vm:~/cloud$ mpirun -np 10 -f machinefile ./s2
Hello world from processor master-vm, rank 5 out of 10 processors
Hello world from processor master-vm, rank 0 out of 10 processors
Hello world from processor slave5-vm, rank 4 out of 10 processors
Hello world from processor slave2-vm, rank 6 out of 10 processors
Hello world from processor slave2-vm, rank 1 out of 10 processors
Hello world from processor slave5-vm, rank 9 out of 10 processors
Hello world from processor slave3-vm, rank 7 out of 10 processors
Hello world from processor slave4-vm, rank 3 out of 10 processors
Hello world from processor slave3-vm, rank 2 out of 10 processors
Hello world from processor slave4-vm, rank 8 out of 10 processors
vagrant@master-vm:~/cloud$ |
```

Figura 42: Lanzamiento de Cluster en Marcha. Prueba de Conectividad incluyendo a todas las máquinas del cluster.

```
vagrant@master-vm:~/cloud$ time mpirun -np 10 -f machinefile ./s2
Hello world from processor master-vm, rank 0 out of 10 processors
Hello world from processor master-vm, rank 5 out of 10 processors
Hello world from processor slave4-vm, rank 3 out of 10 processors
Hello world from processor slave3-vm, rank 7 out of 10 processors
Hello world from processor slave3-vm, rank 2 out of 10 processors
Hello world from processor slave4-vm, rank 8 out of 10 processors
Hello world from processor slave2-vm, rank 1 out of 10 processors
Hello world from processor slave5-vm, rank 4 out of 10 processors
Hello world from processor slave2-vm, rank 6 out of 10 processors
Hello world from processor slave5-vm, rank 9 out of 10 processors

real    0m1.001s
user    0m0.212s
sys     0m0.048s
vagrant@master-vm:~/cloud$
```

Figura 43: Lanzamiento de Cluster en Marcha. Pruebas con time.

```
connection to 127.0.0.1:2204 failed.  
PS D:\PERSONAL\UTP\FichaTecnica-ProyectoGrado\ClusterVagrant\ClusterMPICH\Version1> vagrant destroy -f  
==> slave5-vm: Forcing shutdown of VM...  
==> slave5-vm: Destroying VM and associated drives...  
==> slave4-vm: Forcing shutdown of VM...  
==> slave4-vm: Destroying VM and associated drives...  
==> slave3-vm: Forcing shutdown of VM...  
==> slave3-vm: Destroying VM and associated drives...  
==> slave2-vm: Forcing shutdown of VM...  
==> slave2-vm: Destroying VM and associated drives...  
==> master-vm: Forcing shutdown of VM...  
==> master-vm: Destroying VM and associated drives...  
PS D:\PERSONAL\UTP\FichaTecnica-ProyectoGrado\ClusterVagrant\ClusterMPICH\Version1> |
```

Figura 44: Destruccion de entorno de Cluster.

10. Resultados

Luego de haber realizado todo el montaje y despliegue del cluster MPI utilizando Vagrant el tiempo de instalación y configuración utilizado en el proceso se reduce significativamente comparándolo al método manual, a continuación la siguiente tabla refleja todo los pasos requeridos de una sola maquina :

#	Metódo	Descripción	Tiempo
1	Manual	Creación y configuración de parámetros de Maquina Virtual	12
2	Manual	Descarga de imagen de Sistema Operativo Ubuntu LTS16.04	10
3	Manual	Instalación de Sistema Operativo Ubuntu LTS 16.04 y usuarios	15-20
4	Manual	Configuración de Direcccionamiento IP y archivos de configuración ssh	10
5	Manual	Creación de Usuario a utilizar por el cluster	6
6	Manual	Descarga de Archivos Binarios MPICH y Compilación	8
7	Manual	Instalación y Configuración de Sistema de Archivos NFS	6
8	Manual	Exportación de Imagen .OVA como imagen base para nodos esclavos	15
9	Manual	Importación de Imagen base .OVA y creación de Maquinas Virtuales por cada nodo	18
10	Manual	Pruebas de conectividad y ejecución con MPICH	10
Tiempo Total Estimado de creación de Cluster, minutos			110

La estimación anterior de 110 minutos, se obtiene luego de haber realizado las múltiples actividades durante varias veces y realizar una toma de tiempos utilizando un cronometro. Al realizar este mismo proceso con vagrant el tiempo de creación e instalación de un solo nodo toma aproximadamente 6 minutos, siempre la primera vez que instanciamos el cluster, Vagrant debe realizar la descarga de la imagen base custom creada para el proyecto <https://app.vagrantup.com/c1b3rh4ck>, el tiempo aproximado de esta descarga es de 6 minutos con una conexión mínima de 10Mbps, el tamaño de la imagen tiene un total de aproximadamente 500MB, Vagrant Cloud utiliza Amazon Web Service, S3 para realizar el delivery de la imagen.

Al realizar la revisión de las actividades para inicializar el cluster se tiene un tiempo estimado como el que se muestra en la siguiente tabla :

#	Metódo	Descripción	Tiempo[min]
1	Automatico	<code>vagrant up</code> , primera vez.Descarga de imagen vagrantcloud.	6-10
2	Automatico	Vagrant Provisioning nodo Maestro	6-8
3	Automatico	Vagrant provisioning por cada nodo esclavo	6-8
Tiempo Total Estimado de creación de Cluster MPI			26

Los 26 minutos incluyen la descarga de la imagen base, pero si esta ya ha sido instanciada al menos una vez restaremos tendremos un tiempo de aprovisionamiento y creación de 16 minutos o menos.

11. Participantes

- Héctor Fabio Jiménez Saldarriaga, Estudiante de pregrado en Ingeniería De Sistema Y Computación: Planear, ejecutar, documentar y evaluar el proyecto.
- Ramiro Andrés Barrios Valencia, MSc. Ingeniería de Sistemas y Computación, Especialista en Electrónica Digital: Director y Asesor del proyecto.

12. Recursos y Presupuesto

12.1. Recursos Humanos

- Director de proyecto.

12.2. Recursos Físicos

- Laboratorio Sirius, CIDT.
Ingeniería de Sistemas. Universidad Tecnológica de Pereira.
- Grupo de Investigación Sirius.
Ingeniería de Sistemas y Computación. Universidad Tecnológica de Pereira.
- clúster Lovelace, Bloque L.
- Computador Tesista, Servidores Físicos a disposición tesista.
- Locaciones Grupo de Investigación Sirius.
Ingeniería de Sistemas y Computación. Universidad Tecnológica de Pereira
- Biblioteca. Universidad Tecnológica de Pereira

A continuación un valor estimado para los recursos considerados en el proyecto:

Recurso Humano	Costo(Peso Colombiano) / Hora	Número Horas	Total	Fuente Financiadora
Investigador (Tesisista)	6.500	200	1.300.000	Universidad
Director / Asesor	38.000	80	3.040.000	Universidad
Compra o Alquiler de maquinaria y Equipos	Costo(Peso Colombiano)	Cantidad	Total	Fuente Financiadora
Computador Thinkpad X1 6TH, procesador i7 8GEN, 1TB DD y 16GB RAM	8.100.000	1	8.100.000	Tesisista
Fungibles	Costo(Peso Colombiano)	Cantidad	Total	Fuente Financiadora
Papelería y otros	50.000		50.000	Tesisista
Servicios públicos	130.000		130.000	Tesisista
Costo Total del Proyecto			12.620.000	

13. Cronograma

A continuación se describe un esquema de las actividades que se desarrollarán durante el proyecto, cada una con su respectiva duración:

Cronograma de Actividades						
Actividad / Tiempo(Meses)	1	2	3	4	5	6
Consultas e investigación sobre MPI, clúster y computación de alto rendimiento.	X					
Investigación y Consulta de Infraestructura como Código, entendimiento del funcionamiento interno de Vagrant	X					
Pruebas de aprovisionamiento automático de diferentes elementos de infraestructura con Vagrant	X					
Implementación de algoritmos y scripts para la instalación de utilidades		X				
Pruebas de funcionamiento comunicación master-nodos			X			
Ejecución de algoritmo en clúster				X		
Elaboración de informe final					X	
Entrega						X

14. Conclusiones

- Hacer uso de herramientas de IaC como Vagrant nos permite aprovisionar, gestionar infraestructuras TI de cualquier tipo a través del uso de código fuente de manera que las operaciones de configuración, instalación son mas rápidos, seguros y consistentes como se observa en el capítulo de implementación 9.
- Los hipervisores de maquinas virtuales son herramientas importantes en el área de ingeniería de sistemas pues permiten realizar pruebas y realizar simulación de la complejidad que se experimenta a gran escala de cluster de computación de alto desempeño. También hace posible que el futuro ingeniero, pueda experimentar en la configuración, aprovisionamiento e implementación de productos de software de la industria.
- Como pudo observarse en el capítulo 9 De manera *declarativa* es posible manipular todos los componentes del cluster MPI, componentes de networking, componentes de configuración de maquinas virtuales, e instalaciones durante el aprovisionamiento en los nodos. Esta manera declarativa nos permite tener resultados completamente idempotentes asegurando un estado final de los componentes a diferencia de lo que se observaría en infraestructuras virtuales clásicas que se realiza de manera imperativa.
- Los Estudiantes de la clase de Computación de Alto Desempeño ó HPC pueden realizar el uso de este cluster para enfocarse mas en el entendimiento de como funciona el paralelismo en aplicaciones e implementaciones de código. Además de poder experimentar también con la herramienta Vagrant.

15. Anexo A: VagrantFile Cluster HPC MPI, Multinucleo Multimaquina

```
#hfjimenez@utp.edu.co, VagrantFile proyecto de Grado
#url: https://github.com/h3ct0rjs/FichaTecnica-ProyectoGrado/
SLAVES_NUM = (ENV['SLAVES_NUM'] || 2).to_i
SLAVE_CORE = (ENV['SLAVE_CORES'] || 1).to_i
SLAVE_MEMORY = "384"
MASTER_CORES = ( ENV['SLAVE_CORES'] || 2 ).to_i
MASTER_MEMORY = "1024"
GUI_P = false

Vagrant.configure("2") do |config|
  config.vm.define "master-vm" do | vm1|
    vm1.vm.hostname = "master-vm"
    vm1.vm.box = "c1b3rh4ck/cluster-mpi1804"
    vm1.vm.box_check_update = false
    vm1.vm.provider "virtualbox" do |vb|
      vb.gui = GUI_P
      vb.memory = MASTER_MEMORY
      vb.cpus = MASTER_CORES
    end

    vm1.vm.network "private_network", ip: "192.168.33.10"
    scriptmaster.sub! 'SLAVES_NUM', SLAVES_NUM.to_s
    vm1.vm.provision "shell", inline: scriptmaster.sh
  end

  1.upto(SLAVES_NUM) do |i|
    config.vm.define "slave#{i}-vm" do | slave|
      slave.vm.box = "c1b3rh4ck/cluster-mpi1804"
      slave.vm.hostname = "slave#{i}-vm"
      slave.vm.box_check_update = false

      slave.vm.provider "virtualbox" do |vb|
        vb.gui = GUI_P
        vb.memory = SLAVE_MEMORY
        vb.cpus = SLAVE_CORE
      end
      slave.vm.network "private_network", ip: "192.168.33.2#{i}"
      slave.vm.provision "shell", inline: scriptmaster.sh
    end
  end
end
```

16. Bibliografía

- [1] *Una guía para el conocimiento de Scrum (Guía SBOKTM)*, 2013 ed. VMEdU, Inc, 2013.
- [2] *What is a Virtual Machine, RedHat Official Website*, 2020 (accessed February 3, 2020).
- [3] *What is a Hypervisor, RedHat Official Website*, 2020 (accessed February 5, 2020).
- [4] What is devops?, amazon web services. url<https://aws.amazon.com/devops/what-is-devops> Oct 4 ,2019.
- [5] AMBIT. Devops: Qué es y su poder en la gestión de servicios it. url<https://www.ambit-bst.com/blog/devops-que-es>: :text=El2018.
- [6] AMBIT. Press release, oracle buys sun. url<https://www.oracle.com/corporate/pressrelease/oracle-buys-sun-042009.html>, 2018.
- [7] ATlassian. What is git? url<https://www.atlassian.com/git/tutorials/what-is-git>, 2018.
- [8] BEYER, B., JONES, C., PETOFF, J., AND MURPHY, N. R. *Site Reliability Engineering: How Google Runs Production Systems*, 1st ed. O'Reilly Media, Inc., 2016.
- [9] COTER, S. Oracle vm virtualbox: Networking options and how-to manage them. url<https://blogs.oracle.com/scoter/networking-in-virtualbox-v2>, 2017.
- [10] CRISTINA, J. R. El legendario origen del movimiento devops. url<https://www.paradigmadigital.com/techbiz/el-legendario-origen-del-movimiento-devops> 2016.
- [11] CRISTINA, J. R. Qué es devops (y sobre todo qué no es devops). url<https://www.paradigmadigital.com/techbiz/que-es-devops-y-sobre-todo-que-no-es-devops> 2016.

- [12] DOCS, M. Aprovisionar máquinas virtuales en el tejido de vmm.
url<https://docs.microsoft.com/es-es/system-center/vmm/provision-vms?view=sc-vmm-2017>. 2017.
- [13] DOCS., V. Chapter 6. virtual networking.
url<https://www.virtualbox.org/manual/ch06.html>, 2020.
- [14] DOCS, V. N. Chapter 6. virtual networking.
url<https://www.virtualbox.org/manual/ch06.html>, 2020.
- [15] DUSELIS, J. U., CAUICH, E. E., WANG, R. K., AND SCHERSON, I. D. Resource selection and allocation for dynamic adaptive computing in heterogeneous clusters. In *2009 IEEE International Conference on Cluster Computing and Workshops* (Aug 2009), pp. 1–9.
- [16] GEEKSFORGEEKS. Operating system — difference between multitasking, multithreading and multiprocessing.
url<https://www.geeksforgeeks.org/difference-between-multitasking-multithreading-and-multiprocessing/>. 2018.
- [17] HASHIMOTO, M. *Vagrant: Up and Running Create and Manage Virtualized Development Environments*, 1st ed. O’Reilly Media, Inc., 2013.
- [18] INTERNET ENGINEERING TASK FORCE, T. HAYNES, E. Nfsv4: Network file system protocol specification,. url<https://tools.ietf.org/html/rfc7530>, 2015.
- [19] KHAWAJA, G. Linux hardening: A 15-step checklist for a secure linux server.
url<https://www.pluralsight.com/blog/it-ops/linux-hardening-secure-server-checklist>, 2017.
- [20] KUMBHAR, P., HINES, M., OVCHARENKO, A., A. MALLON, D., KING, J., SAINZ, F., SCHÜRMANN, F., AND DELALONDRE, F. Leveraging a cluster-booster architecture for brain-scale simulations.
- [21] LATINOAMERICA, H. P. Definiciones principales ¿que es la infraestructura como codigo?. url<https://www.hpe.com/lamerica/es/what-is/infrastructure-as-code.html>, 2019.
- [22] MAS, R. H. . R. *El libro del administrador de Debian*, 1st ed. Freexian SARL, 2012.
- [23] MORRIS, K. *Infrastructure as Code: Managing Servers in the Cloud*, 1st ed. O’Reilly Media, Inc., 2016.

- [24] MÚNERA, A. M., AND BEDOYA, J. W. B. Implementación de un cluster homogéneo para la resolución de problemas de alta complejidad computacional. *Avances en Sistemas e Informática; Vol. 5, núm. 3 (2008); 41-48 Avances en Sistemas e Informática; Vol. 5, núm. 3 (2008); 41-48 1909-0056 1657-7663 5, 3 (2008), 41–48.* Derechos de autor reservados.
- [25] NETWORKING GROUP SUN MICROSYSTEMS, I. Nfs: Network file system protocol specification. [urlhttps://tools.ietf.org/html/rfc1094](https://tools.ietf.org/html/rfc1094), 1989.
- [26] OFFICE, T. U. A. I. S. Red hat enterprise linux 7 hardening checklist. [urlhttps://security.utexas.edu/os-hardening-checklist/linux-7](https://security.utexas.edu/os-hardening-checklist/linux-7), 2019.
- [27] REDHAT. Automation, what is provisioning? [urlhttps://www.redhat.com/en/topics/automation/what-is-provisioning](https://www.redhat.com/en/topics/automation/what-is-provisioning), 2020.
- [28] ROUSE, M. Sistema de archivos de red, nfs. [urlhttps://searchdatacenter.techtarget.com/es/definicion/Sistema-de-archivos-de-red-NFS](https://searchdatacenter.techtarget.com/es/definicion/Sistema-de-archivos-de-red-NFS) 2017.
- [29] SINGH, H. Roadway to it revolution: The history of devops. [urlhttps://www.appknox.com/blog/history-of-devops](https://www.appknox.com/blog/history-of-devops), Oct 4 ,2019.
- [30] STACKOVERFLOW. Failed to open/create the internal network vagrant on windows10. [urlhttps://stackoverflow.com/questions/33725779/failed-to-open-create-the-internal-network](https://stackoverflow.com/questions/33725779/failed-to-open-create-the-internal-network) 2015.
- [31] THEO UNGERER, B. R., AND SILC, J. *A Survey of Processors with Explicit Multithreading*, 1st ed. March 2003.
- [32] TITHINK. Metodologías ágiles ¿qué son y para qué sirven? [urlhttps://www.tithink.com/es/2018/10/16/metodologias-agiles-que-son-y-para-que-sirven](https://www.tithink.com/es/2018/10/16/metodologias-agiles-que-son-y-para-que-sirven) 2018.
- [33] TRIMSTRAY. linux-hardening-checklist. [urlhttps://github.com/trimstray/linux-hardening-checklist](https://github.com/trimstray/linux-hardening-checklist) 2019.
- [34] TUCKER, C., SHUFFELTON, D., JHALA, R., AND LERNER, S. Opium: Optimal package install/uninstall manager. In *Proceedings of the 29th International Conference on Software Engineering (USA, 2007), ICSE '07*, IEEE Computer Society, p. 178–188.

- [35] UDG. Multiprocesamiento. urlhttps://www.udg.co.cu/cmap/sistemas_operativos/sistemas_operativos
- [36] VILLAN, V. R. Las metodologías ágiles más utilizadas y sus ventajas dentro de la empresa. url<https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/>, 15 MARZO, 2019.
- [37] vSPHERE DOCS, V. Aprovisionar máquinas virtuales en el tejido de vmm. url<https://docs.vmware.com/es/VMware-vSphere/6.5/com.vmware.vsphere.vmm.admin.doc/E19DA34B-B227-44EE-B1AB-46B826459442.html>, 2019.
- [38] XSEDE. A brief history of high-performance computing (hpc). url<https://confluence.xsede.org/pages/viewpage.action?pageId=1677620>.