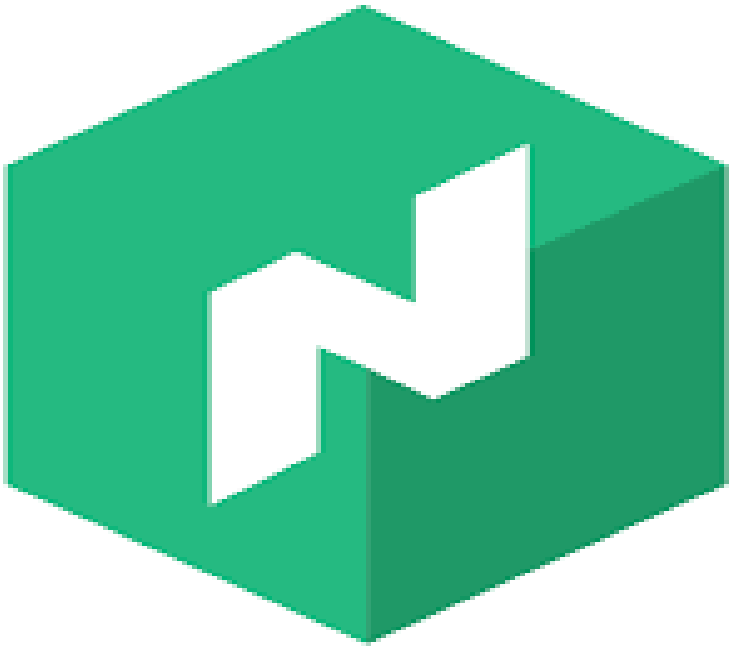
Orquestación Docker Con Nomad Hashicorp



HashiCorp

Nomad

I.E.S Gonzalo Nazareno

Nombre Del Alumno/a: Kevin Ariza García

Curso: 2ºASIR

Módulo: Proyecto y Formación en Centros de Trabajo (FCT)

Nombre/s Del/Los Profesor/es: Alberto Molina Coballes, José Rafael Luque Giráldez, José Domingo Muñoz Rodriguez, Raúl Ruiz Padilla

ÍNDICE DEL CONTENIDO

1. Introducción Del Proyecto.

Mi nombre es Kevin Ariza García, y soy alumno de segundo año en el Ciclo Formativo de Grado Superior de Informática en **“Administración de Sistemas Informáticos y Redes”,** y voy a explicar en que va a consistir mi proyecto de Fin de Ciclo, para la asignatura o módulo profesional de **“FCT”.**

Mi proyecto principalmente consistirá, en la orquestación de varios contenedores de **“Docker”,** para poder realizar el despliegue de varias aplicaciones como **“microservicios”,** en una o varias máquinas o equipos a utilizar para el proyecto a realizar en cuestión, donde se explicará más adelante cuando se vaya a presentar el escenario a utilizar para el proyecto. También, veremos la gestión y monitorización de los contenedores orquestados, para el despliegue de una aplicación con **Nomad Hashicorp,** mediante la consola gráfica **“Web-UI”** de la herramienta descrita.

Dicha orquestación de contenedores **“Docker”,** se van a realizar mediante la aplicación o herramienta llamada **“Nomad Hashicorp”,** donde dicha herramienta está desarrollada y distribuida por la compañía Hashicorp. Comentar que la herramienta descrita, es una herramienta similar a Kubernetes, donde cada herramienta tiene funciones similares entre sí, pero que a la hora de la verdad, no llegan a ser lo mismo ambas herramientas.

Básicamente, los apartados que se van a tratar en el proyecto a presentar y realizar, van a ser algunos conceptos sobre la herramienta que se va a utilizar para la realización del proyecto, la instalación y configuración de dicha herramienta, mostrando los contenedores **Docker** orquestados de la aplicación desplegada, y poder ver las diferentes opciones de la gestión y monitorización de dicha orquestación de contenedores **Docker**, mediante la consola **“Web-UI”** de **Nomad Hashicorp.**

Mi objetivo principal con el desarrollo del proyecto, es poder dar a conocer una herramienta similar a **“Kubernetes”**, para el despliegue u orquestación de contenedores **“Docker”**, hacia uno o varios equipos o nodos servidores, y también, poder llevar el control de la gestión y monitorización de la orquestación de contenedores **Docker**, de una forma sencilla y gráfica.

Partiendo desde éste punto, todo lo que viene a continuación, será el comienzo del desarrollo del proyecto a realizar, que está dividido en varios apartados, donde en cada uno de los apartados, se explicarán todos los conceptos de una forma que se pueda entender fácilmente. Sin más dilación, vamos manos a la obra.

2. ¿Qué Es Nomad Hashicorp?

Nomad Hashicorp es una aplicación o herramienta flexible, destinada a la orquestación de contenedores y equipos virtuales, que permite a una organización o empresa, poder implementar y administrar fácilmente cualquier aplicación en un contenedor, equipo virtual o heredada, utilizando un flujo de trabajo unificado, es decir, es un un flujo de trabajo único, ejecutando una carga de trabajo diversa de las aplicaciones, y no en contenedores, microservicios y por lotes.

3. ¿Cómo Funciona Nomad Hashicorp?

Nomad Hashicorp es una herramienta para poder administrar un grupo de máquinas, y poder ejecutar aplicaciones en cada una de las máquinas a crear o creadas. Para ello, extraerá las máquinas y ubicación de las aplicaciones a ejecutar en dicha máquina, permitiendo al usuario declarar lo que quiere ejecutar, donde luego **Nomad Hashicorp** se encargará de controlar la ubicación de ejecución de la aplicación, y la forma en la que se va ejecutar dicha aplicación.

La herramienta **“Nomad Hashicorp”,** está orientado para ser instalado en equipos servidores, donde pueden actuar como **“Servidor”,** para poder gestionar y supervisar los despliegues de contenedores **“Docker”,** o actuar como **“Cliente”,** para poder realizar el alojamiento de los despliegues realizados en el equipo que actúa como **“Servidor”.**

4. Características De Nomad Hashicorp.

La herramienta **“Nomad Hashicorp”** presenta tener una serie de características, las cuales se van a reflejar dichas características a continuación:

- **Soporte de Docker:** Nomad admite contenedores **“Docker”** como tipo de carga de trabajo en primera clase. Los trabajos enviados a Nomad Hashicorp, pueden usar un controlador de ventana acoplable, para poder implementar fácilmente aplicaciones de contenedores en un clúster, aplicando restricciones especificadas por el usuario en concreto, haciendo que la aplicación se ejecute en cada región, centro de datos y entornos de hosts concretos, especificando el número de instancias a ejecutar, donde Nomad Hashicorp luego se encargará de levantar o crear dichas instancias especificadas.

- **Operaciones Simples:** Nomad Hashicorp se envía como un único binario, tanto para los Servidores como para los Clientes, sin necesidad de requerir servicios externos para la coordinación o almacenamiento, combinando características de los gestores de recursos y programadores en un solo sistema. Nomad Hashicorp se construye bajo **"Serf"** y **"Cónsul",** que son herramientas distribuidas por la empresa de **"Hashicorp".**

- **Multi-Datacenter y Multi-Region:** Nomad Hashicorp es capaz de modelar la infraestructura de clúster de contenedores, como grupos de centros de trabajos que forma parte de una región más grande. La programación se opera a nivel regional, permitiendo la programación de centros de trabajos cruzados. Las múltiples regiones federadas se juntan, permitiendo que los trabajos sean registrados regionalmente.

- **Cargas De Trabajo Flexibles:** Nomad Hashicorp tiene soporte extensible para controladores de tareas, permitiéndole ejecutar aplicaciones en contenedores, máquinas virtualizadas y de forma independiente respecto a contenedores y equipos virtuales. Los usuarios pueden iniciar fácilmente contenedores **Docker,** máquinas virtuales o tiempos de ejecución de aplicaciones, como puede ser por ejemplo Apache o Java. Nomad Hashicorp es compatible con sistemas operativos **"Microsoft Windows"**, **"GNU/Linux"**, **"BSD"** y **"OSX",** proporcionando flexibilidad para la ejecución de cualquier carga de trabajo.

- **Diseñado Para Ser Escalado:** Nomad Hashicorp fue diseñado desde cero, para poder soportar infraestructura de escala global, donde Nomad Hashicorp se distribuye y está altamente disponible, utilizado tanto en la elección del líder como la replicación del estado para poder proporcionar disponibilidad ante fallos que puedan ocurrir. Nomad Hashicorp es optimista al mismo tiempo, permitiendo todo los servidores poder participar en las decisiones de programación, aumentando el rendimiento total y reduciendo la latencia de soporte de cargas de trabajos exigentes.

Nota de Curiosidad: A lo largo del desarrollo de la herramienta **"Nomad Hashicorp",** se ha podido comprobar que Nomad Hashicorp se ha adaptado a tamaños de clústers, que superan a la cantidad de **"10000 nodos"** en entornos de producción reales.

Decir que se han reflejado algunas de las características, que presenta tener la herramienta **"Nomad Hashicorp",** donde en realidad presenta tener más características de las que se han reflejado o explicado. Pero he visto conveniente, poder definir las características que he visto que van a ser de más utilidad para todos los usuarios.

5. Ventajas e Inconvenientes De Uso De Nomad Hashicorp.

La herramienta **"Nomad Hashicorp"**, presenta tener una serie de ventajas e inconvenientes, a la hora del uso que vayamos a realizar con dicha herramienta descrita. Dichas ventajas e inconvenientes, se van a reflejar a continuación.

• Ventajas De Nomad Hashicorp.

- Integra una herramienta llamada **"Consul",** que es una herramienta también propia de la compañía **"Hashicorp",** que actúa como una malla de servicios distribuidos para poder conectar, asegurar y configurar servicios, en cualquier plataforma de tiempo de ejecución en nube pública o privada.

- Tiene la capacidad de poder utilizar algunas aplicaciones y servicios, como pueden ser por ejemplo **"Nginx"**, **"HAProxy"** y **"Fabio"**, para poder realizar la automatización de configuración, a través de los datos de **"Consul"** con **"consul-template"**.

- Integra una herramienta llamada **"Vault"**, que es una herramienta también propia de la compañía **"Hashicorp",** donde se encarga de asegurar, almacenar y controlar el acceso de tokens, contraseñas, certificados, claves cifradas para la protección de secretos y otros datos confidenciales, mediante una **interfaz de usuario**, **CLI** o **API HTTP.**

• Inconvenientes De Nomad Hashicorp.

- No es un Servidor para el descubrimiento de resolución de ubicaciones, para las aplicaciones que ya están desplegadas.

- No tiene la capacidad de poder funcionar como balanceador de carga, y tampoco como **"Proxy"** para la redirección de peticiones, entre los diferentes nodos que compone tener un clúster de contenedores **"Docker".**

- No tiene la disposición para la gestión de secretos de una orquestación de contenedores **"Docker".**

6. Comparativa Nomad Hashicorp Con Software Similar.

Nomad Hashicorp es una herramienta como bien se ha dicho anteriormente, para poder orquestar varios contenedores y máquinas virtuales de diferentes tipos, donde pueden tener una o varias aplicaciones desplegadas.

Pero cabe destacar, que en el mundo de la informática, hay herramientas que siempre serán similares unas herramientas con otras, con la intención de poder mejorar funcionalidades que otras herramientas no tienen disponible o pueden ocasionar fallos en dicha herramientas, hacer una innovación que nadie haya hecho, o poder adaptarlos a espacios de trabajos específicos o generales, para el uso de dicha herramienta.

Como nos podemos esperar, **"Nomad Hashicorp"** tiene dura competencia en la orquestación de contenedores, siendo **"Kubernetes"** la más implantada en las empresas, donde poco a poco, está haciendo hueco en el mercado.

A continuación, voy a dejar una comparativa de **"Nomad Hashicorp"** contra tres herramientas de otras que hay disponibles, para la orquestación de contenedores **"Docker".**

• Nomad Hashicorp VS Kubernetes.

Kubernetes es un sistema de orquestación de contenedores, que ha sido diseñado originalmente por la empresa **"Google",** pero tiempo después, pertenece a la asociación **Cloud Native Computing Foundation (CNCF).** Su objetivo principal es proporcionar todas las funciones necesarias, para la ejecución de aplicaciones con **"Docker"** o **"Rkt"** entre otros.

El objetivo principal de la herramienta **"Nomad Hashicorp",** es poder proporcionar administración y programación de clústeres, diseñado con la filosofía de **"UNIX",** componiéndose de herramientas como **"Consul" y "Vault".**

Cuando **"Kubernetes"** solo se centra en la orquestación de contenedores **"Docker",** la herramienta **"Nomad Hashicorp"** tiene un propósito más general, como el poder admitir aplicaciones tanto virtualizadas, en contenedores y de forma autónoma, donde en ellas también se incluye **"Docker".**

Kubernetes está diseñado como una colección de más de media docena de servicios interoperativos, proporcionando una funcionalidad compleja. La coordinación y almacenamiento se proporciona por **"etcd"** en el núcleo, con un estado envuelto en una API consumida por otros servicios que proporcionan API de nivel superior, para funciones de programación.

Kubernetes admite la ejecución en una configuración de alta disponibilidad, pero su configuración es compleja respecto a **"Nomad Hashicorp".**

A diferencia de **"Kubernetes",** la arquitectura de orquestación de contenedores con **"Nomad Hashicorp"** es más simple, siendo de binario único tanto para Servidores como para Clientes, sin tener que requerir de Servidores externos para la coordinación o almacenamiento.

Nomad Hashicorp combina un administrador de recursos ligero y un programador sofisticado en un solo sistema. Se distribuye, está altamente disponible y es operativamente simple.

El servicio de **"Kubernetes"** puede admitir clústeres de más de **"5000 nodos",** mientras que **"Nomad Hashicorp",** puede admitir clústeres de más de **"10000 nodos"** en entornos de producción reales.

Ambos admiten configuraciones **"multi-datacenter"** y **"multi-regions".**

- **Conclusión Comparativa: Kubernetes** está ambientado más para poder orquestar contenedores **"Docker"** en un entorno de trabajo complejo, y donde se vaya a trabajar en nube, mientras que **Nomad Hashicorp,** está orientado más para poder trabajar con equipos Servidores internos de una empresa o entidad, donde no se vaya a consumir altos recursos. Además, **"Kubernetes"** solo puede realizar orquestación de contenedores **"Docker",** mientras que **"Nomad Hashicorp"** puede orquestar diferentes tipos de contenedores y máquinas virtuales. Pero a la hora de la verdad, **"Kubernetes"** es una opción más ideal que **"Nomad Hashicorp",** debido a que presenta tener diferentes herramientas de despliegue de aplicaciones y orquestación de contenedores **"Docker",** para poder llevar una mejor gestión de dichas aplicaciones desplegadas en contenedores. Pero no debemos olvidar, que la cantidad de nodos que podemos tener en **"Nomad Hashicorp",** es mayor que en **"Kubernetes",** donde pueden estar dividido en varios clústers dichos contenedores.

- **Nomad Hashicorp VS Docker Swarm.**

Docker Swarm es la solución nativa de clúster para contenedores **“Docker”,** proporcionando una API compatible con la API remota de **“Docker”,** permitiendo que los contenedores sean programados en varias máquinas.

Docker Swarm solo se utiliza para la ejecución de contenedores **“Docker”,** mientras que **“Nomad Hashicorp”** realiza un uso más general, al igual que se ha comentado con **“Kubernetes”,** haciendo que no únicamente se pueda realizar la ejecución en contenedores **“Docker”,** sino que se puede realizar orquestación de máquinas virtuales también.

Docker Swarm proporciona compatibilidad de API con su API remota, centrándose en la abstracción de los contenedores. Mientras que **“Nomad Hashicorp”,** utiliza una abstracción de trabajos de alto nivel, donde dichos trabajos contienen grupos de tareas que permiten que las aplicaciones más complejas, se expresen y administren fácilmente.

Las arquitecturas de **“Nomad Hashicorp”** y **“Docker Swarm”** difieren entre sí.

Nomad Hashicorp no depende de sistemas externos para la coordinación o el almacenamiento, teniendo gran disponibilidad y compatible con configuraciones **“multi-datacenter”** y **“multi-region”**

Docker Swarm no está distribuido o altamente disponible de forma predeterminada. Tampoco es compatible con configuraciones **“multi-region”** de aislamiento de fallas o federación.

Cuando la replicación está habilitada en **“Docker Swarm”**, usa un modelo **activo / en espera**, haciendo que los otros Servidores no se utilizan para tomar decisiones de programación.

- **Conclusión Comparativa: Docker Swarm** es una herramienta nativa de **“Docker”,** para poder llevar la gestión de sus contenedores. Pero a la hora de la orquestación de máquinas u contenedores, **Nomad Hashicorp** veo una mejor opción para poder ser usado, ya que es capaz de orquestar contenedores y equipos virtuales de diferentes tipos. Pero si quieres orientarte solo a contenedores **“Docker”,** veo mejor opción **“Docker Swarm”.**

• Nomad Hashicorp VS Terraform

Terraform es una herramienta, que sirve para poder construir, cambiar y versionar infraestructura de manera segura y eficiente. La ejecución de **Terraform** está orientada para el despliegue de una sola aplicación o centro de datos completo, haciendo que genere un plan de ejecución descrito, para luego ser ejecutado para la construcción de una infraestructura definida.

A medida que la configuración vaya cambiando o modificándose con **"Terraform", Terraform** puede determinar que cambios se han producido y poder crear planes de ejecución incrementales que se pueden aplicar.

Terraform está diseñado para admitir cualquier tipo de recurso, sea de alto o bajo nivel, como pueden ser instancias de computación, almacenamiento, redes, entradas DNS, características de SaaS, etc.

Terraform sabe como crear, aprovisionar y gestionar el ciclo de vida de estos recursos.

Nomad Hashicorp se ejecuta en la infraestructura existente y administra el ciclo de vida de las aplicaciones que se ejecutan en esa misma infraestructura.

Terraform es una herramienta fuera de línea, donde se ejecuta hasta su finalización. Mientras que **Nomad Hashicorp**, es un sistema en línea con servidores de larga duración.

Nomad Hashicorp permite que se envíen nuevos trabajos, se actualicen o se eliminen trabajos existentes, pudiendo manejar las fallas aparecidas o existentes en los nodos. Todo éste proceso, requiere que sea disparado continuamente, mientras que **"Terraform"** hace todo el proceso en un solo disparo.

Para infraestructuras pequeñas, **Nomad Hashicorp** se ejecuta de mejor forma, ya que administra aplicaciones de programación a máquinas de forma dinámica. Mientras que **Terraform,** se utiliza en escalas más grandes, para administrar aplicaciones de programación a máquinas de forma estática.

- **Conclusión Comparativa:** Si quieres desplegar una sola aplicación de forma estática, donde se realiza un único despliegue, y que vaya a estar orientado a infraestructuras de gran escala, **"Terraform"** es más conveniente usarlo. Pero si queremos desplegar varias aplicaciones, donde se tienen que realizar despliegue continuamente y en infraestructuras pequeñas, **"Nomad Hashicorp"** es la mejor opción para ello.

Hay más herramientas para la orquestación de contenedores **"Docker"** que existen hoy en día en el mercado de la informática, pero como se ha visto en las comparativas, se han seleccionado las herramientas que más se utilizan y se implementan hoy en día en las empresas

- ## ¿Cuál Es La Finalidad Principal De Las Comparativas?

La finalidad principal con las comparativas presentadas anteriormente, es poder saber en qué situaciones debemos utilizar una herramienta de despliegue y orquestación de contenedores, ya puede ser por ejemplo **"Docker",** dependiendo de las situaciones que se presenten, viendo las posibles ventajas que pueden otorgar cada herramienta, contado con sus inconvenientes también, claro está. Por ello, no se hace para dejar una herramienta en mejor posición que otra, sino para dar las posibles diferencias y similitudes que presentan tener cada herramienta, para saber elegir la más adecuada y adaptada a las necesidades del desarrollador y administrador de sistemas.

7. ¿Cuando Se Debería Utilizar Principalmente Nomad Hashicorp?

La herramienta **"Nomad Hashicorp"** ha sido diseñada principalmente para la orquestación de contenedores y equipos informáticos, orientados a equipos Servidores de empresas, donde no vayan a trabajar en grandes infraestructuras y tampoco en nubes de grandes infraestructuras.

Pero hay una serie de funcionalidades y características, que hacen que sea lo más apropiado utilizar la herramienta descrita, donde se van a reflejar a continuación, adecuándose a las situaciones a presentarse.

- **Gestion De Contenedores Docker:** Nomad Hashicorp admite un flujo de trabajo **“Docker”** de primera clase, donde se integra a la perfección con **“Consul” y Vault”,** para poder habilitar una solución completa a la vez que maximiza la flexibilidad operativa. **Nomad Hashicorp** es fácil de utilizar, pudiendo escalar miles de nodos en un sólo **“clúster”,** y también se puede implementar fácilmente en centros de trabajos privados y múltiples nubes.

- **Implementación de Aplicaciones Heredadas:** Nomad Hashicorp admite de forma nativa, la ejecución de aplicaciones heredadas, aplicaciones binarias estáticas, aplicaciones JAR y comandos simples de sistema operativo. Las cargas de trabajo se aislan, de forma nativa, en los tiempos de ejecución y se empaquetan, para poder maximizar la eficiencia y utilización, dando a una reducción del costo de dichas aplicaciones.

- **Microservicios:** Nomad Hashicorp se integra elegantemente con **Consul,** para poder realizar el registro automático de servicios y representación dinámica de archivos de configuración. Nomad Hashicorp y Consul juntos, proporcionan una solución ideal para administrar **microservicios,** facilitando la adopción del paradigma.

- **Procesamiento por Lotes de Cargas de Trabajo:** Nomad Hashicorp puede ejecutar de forma nativa, trabajos por lotes, trabajos parametrizados y cargas de trabajo **Spark.** La arquitectura de **Nomad Hashicorp** permite una fácil escalabilidad y una estrategia de programación concurrente y optimista que puede generar miles de implementaciones de contenedores por segundo.

- **Implementación Multi-Region y Multi-Cloud:** Nomad Hashicorp está diseñado para manejar de forma nativa, los centros de datos y las implementaciones de múltiples regiones, y es independiente de la nube. Nomad Hashicorp le permite poder programar en centros de datos privados, que se ejecutan sin restricciones en **OpenStack** o **VMWare,** junto con una implementación en la nube de **AWS, Azure** o **GCE.** Toda la implementación descrita anteriormente, le permite que sea más fácil de migrar las cargas de trabajo de forma incremental, y utilizar la nube para la explosión o despliegue.

8. Descripción Del Escenario A Montar.

El escenario que voy a montar, para poder realizar la orquestación de contenedores **Docker** con **Nomad Hashicorp**, constará de un equipo Servidor virtual creado con **Vagrant**, donde en dicho Servidor, se instalarán las siguientes herramientas:

- **Nomad Hashicorp.**
- **Consul.**
- **Docker.**

Con las herramientas descritas, donde serán instaladas en el equipo Servidor virtual, se realizará la orquestación de contenedores **Docker,** a través de una serie de pasos que se irán explicando a lo largo del proyecto.

El escenario descrito, será montado o creado mediante un fichero de **Vagrantfile**, donde en dicho fichero, se irán configurando los diferentes parámetros para el equipo Servidor virtual a crear, ya sea la instalación de las herramientas descritas, la memoria RAM que tendrá dicho equipo, direccionamiento de red de la máquina, la cantidad de interfaces de red que tendrá, etc.…

El sistema operativo que se utilizará, para poder realizar la orquestación de contenedores **Docker** con **Nomad Hashicorp,** será un sistema operativo **GNU/Linux Ubuntu Bionic 18.04.** Aunque también se puede utilizar otra distribución diferente **GNU/Linux** a la que se ha descrito, pero he decidido utilizar la distribución **Ubuntu** para ésta situación.

Los recursos hardware que tendrá el equipo Servidor a crear, serán de recursos bajos, debido a que el Hardware de mi máquina anfitriona, no presenta tener un Hardware de prestaciones altas, por lo que la experiencia a nivel de rendimiento, no será el mismo que si tuviese un equipo virtual con mejores prestaciones.

En el siguiente apartado, se explicará la creación del escenario descrito, donde se mostrarán los parámetros a configurar para el equipo Servidor virtual a crear.

9. Creación Del Escenario: Nomad Hashicorp.

Vamos a proceder con la creación del escenario que vamos a utilizar, para poder realizar la orquestación de contenedores **Docker** con **Nomad Hashicorp,** en un equipo Servidor virtual **Ubuntu Bionic 18.04.** Para ello, vamos a seguir los siguientes pasos que se mostrarán a continuación.

Primero, situados en la máquina anfitriona, vamos a acceder a la **Terminal** del sistema operativo, y siendo **usuario normal** en la **Terminal** del sistema operativo, vamos a crear una carpeta (donde en mi caso lo he llamado Nomad-Project), y creada la nueva carpeta, vamos a acceder a dicha carpeta creada.

```
usuario@debian:~/vagrant$ mkdir Nomad-Project && cd Nomad-Project
usuario@debian:~/vagrant/Nomad-Project$
```

A continuación, situados dentro de la carpeta anteriormente descrita, vamos a generar un fichero **Vagrantfile** automáticamente, ya que será el fichero que nos servirá, para poder crear el equipo Servidor virtual con **Vagrant.** Para ello, ejecutamos el siguiente comando:

```
usuario@debian:~/vagrant/Nomad-Project$ vagrant init
```

Comprobamos que el fichero **Vagrantfile** se ha generado correctamente:

```
usuario@debian:~/vagrant/Nomad-Project$ ls
Vagrantfile
usuario@debian:~/vagrant/Nomad-Project$
```

Como se puede observar, el fichero de **Vagrantfile** se ha generado correctamente, por lo que ahora, vamos a editar dicho fichero generado, mediante la siguiente instrucción:

```
usuario@debian:~/vagrant/Nomad-Project$ nano Vagrantfile
```

Situados dentro del fichero **Vagrantfile,** vamos a escribir las siguientes líneas a mostrarse en el fichero descrito, donde se irá explicando paso a paso, la configuración adaptada para la creación del equipo servidor virtual:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
```

- Creamos o configuramos un **script,** donde se ejecutará de forma automática, cuando el equipo servidor virtual esté iniciado o ejecutándose en la máquina anfitriona.

```
$script = <<SCRIPT
```

- Primero, se procede con la instalación de **Docker** en el equipo servidor virtual, donde en ella, se siguen una serie de pasos, para poder dejar configurado **Docker** en el equipo Servidor.

```
echo "Instalación De Docker"
# Actualizamos los repositorios de paquetes del sistema operativo en el equipo servidor
sudo apt-get update
sudo apt-get remove docker docker-engine docker.io
# Instalamos paquetes necesarios para la instalación de Docker
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common -y
# Añadimos el repositorio de Docker en los repositorios del paquete del sistema operativo del equipo servidor
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
# Actualizamos de nuevo los repositorio de paquetes y procedemos a instalar Docker en el equipo servidor
sudo apt-get update
sudo apt-get install -y docker-ce
```

```
# Reiniciamos Docker para poder asegurarnos de que obtenemos la última versión estable, en caso de que hubiese actualización
sudo service docker restart
# Añadimos usuario Vagrant al grupo del sistema operativo Docker
sudo usermod -aG docker vagrant
sudo docker --version
```

- A continuación, se instalarán una serie de paquetes requeridos, para poder realizar la instalación de **Nomad Hashicorp y Consul** en el equipo servidor virtual.

```
# Paquetes requeridos para la instalación de Nomad Hashicorp y Consul
sudo apt-get install unzip curl vim -y
```

- En el siguiente paso, se procede con la instalación de **Nomad Hashicorp** en el equipo servidor virtual, donde se sigue los siguientes procedimientos o pasos descritos.

```
echo "Instalación De Nomad Hashicorp"
NOMAD_VERSION=0.9.1
cd /tmp/
# Descargamos y descomprimimos la versión de Nomad Hashicorp indicada en el equipo servidor
curl -sSL https://releases.hashicorp.com/nomad/${NOMAD_VERSION}/nomad_${NOMAD_VERSION}_linux_amd64.zip -o nomad.zip
unzip nomad.zip
# Instalamos Nomad Hashicorp en el equipo servidor
sudo install nomad /usr/bin/nomad
# Creamos una carpeta llamada nomad.d y asignamos permisos de escritura a todos
sudo mkdir -p /etc/nomad.d
sudo chmod a+w /etc/nomad.d
```

- Seguidamente, se procederá con la instalación de **Consul** en el equipo servidor virtual, donde se sigue los siguientes procedimientos o pasos descritos.

```
echo "Instalación De Consul"
CONSUL_VERSION=1.5.1
# Descargamos y descomprimimos la versión de Nomad Hashicorp indicada en el equipo servidor
curl -sSL https://releases.hashicorp.com/consul/${CONSUL_VERSION}/consul_${CONSUL_VERSION}_linux_amd64.zip > consul.zip
unzip /tmp/consul.zip
# Instalamos Consul en el equipo Servidor
sudo install consul /usr/bin/consul
```

- Como paso opcional, se creará una unidad de systemd para **Consul**, con el motivo de poder iniciar, parar y reiniciar el servicio de **Consul,** de una forma más sencilla y mediante una unidad creada en el sistema operativo del equipo servidor virtual.

```
# Creamos una unidad systemd para poder iniciar, parar y reiniciar Consul en el equipo servidor
(
cat <<-EOF
  [Unit]
  Description=consul agent
  Requires=network-online.target
  After=network-online.target

  [Service]
  Restart=on-failure
  ExecStart=/usr/bin/consul agent -dev
  ExecReload=/bin/kill -HUP $MAINPID

  [Install]
  WantedBy=multi-user.target
EOF
) | sudo tee /etc/systemd/system/consul.service
```

- Se habilitará la unidad de systemd creada, y acto seguido iniciamos **Consul** en el equipo servidor virtual.

```
sudo systemctl enable consul.service
sudo systemctl start consul
```

- Proseguidamente, se procederá a instalar **$bin** en el equipo servidor virtual, a través de los siguientes procedimientos o pasos descritos.

```
# Instalar $bin en el equipo servidor

for bin in cfssl cfssl-certinfo cfssljson
do
  echo "Instalación De $bin"
    curl  -sSL  https://pkg.cfssl.org/R1.2/${bin}_linux-amd64  >  /tmp/$
{bin}
  sudo install /tmp/${bin} /usr/local/bin/${bin}
done
nomad -autocomplete-install

SCRIPT
```

- Configuramos el sistema operativo del equipo servidor virtual y el nombre del equipo servidor virtual.

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/bionic64"
  config.vm.hostname = "nomad-server"
  config.vm.provision "shell", inline: $script, privileged: false
```

- Configuramos la memoria RAM que tendrá asociado el equipo servidor virtual, donde en éste caso, tendrá asociado una memoria RAM de **1GB (1024 MB).** También se configurará la interfaz de red que tendrá asociado el equipo servidor virtual, donde en éste caso, tendrá una interfaz de red configurada en **Sólo Anfitrión (Host-Only),** con una dirección IP asociada a dicha interfaz de red del equipo servidor virtual.

```
  # Expose the nomad api and ui to the host
      config.vm.network :private_network, ip: "10.0.200.50"


      # Increase memory for Virtualbox
      config.vm.provider "virtualbox" do |vb|
            vb.memory = "1024"
      end
end
```

Configurado el fichero **Vagrantfile** para la creación del equipo servidor virtual, guardamos los cambios efectuados y salimos de dicho fichero.

En el siguiente paso, vamos a proceder con la creación del equipo servidor virtual con **Vagrant**, mediante el fichero de **Vagrantfile** creado y configurado anteriormente. Para ello, ejecutamos el siguiente comando:

```
usuario@debian:~/vagrant/Nomad-Project$ vagrant up
```

Durante el proceso de creación del equipo servidor virtual con **Vagrant,** se irán ejecutando los pasos o instrucciones definidas en el fichero **Vagrantfile** creado y configurado anteriormente.

```
==> default: Importing base box 'ubuntu/bionic64'...
…
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
…
```

```
==> default: Running provisioner: shell...
    default: Running: inline script
    default: Instalación De Docker
    default: Hit:1 http://archive.ubuntu.com/ubuntu bionic InRelease
      default: Get:2 http://security.ubuntu.com/ubuntu bionic-security
InRelease [88.7 kB]
…
default: Instalación De Nomad Hashicorp
    default: Archive:  nomad.zip
    default:   inflating: nomad
…
    default: Instalación De Consul
    default: Archive:  /tmp/consul.zip
    default:   inflating: consul
    default:   [Unit]
    default:   Description=consul agent
    default:   Requires=network-online.target
    default:   After=network-online.target
    default:
    default:   [Service]
    default:   Restart=on-failure
    default:   ExecStart=/usr/bin/consul agent -dev
    default:   ExecReload=/bin/kill -HUP
    default:
    default:   [Install]
    default:   WantedBy=multi-user.target
             default:   Created   symlink   /etc/systemd/system/multi-
user.target.wants/consul.service → /etc/systemd/system/consul.service.
…
    default: Instalación De cfssl
    default: Instalación De cfssl-certinfo
    default: Instalación De cfssljson
```

Cuando el equipo servidor virtual se haya creado correctamente con **Vagrant,** mediante el fichero de **Vagrantfile** creado y configurado anteriormente, vamos a acceder al equipo servidor virual con **Vagrant,** a través de la siguiente instrucción:

```
usuario@debian:~/vagrant/Nomad-Project$ vagrant ssh
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-45-generic x86_64)

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

  System information as of Mon May 27 11:45:02 UTC 2019

  System load:  0.15              Users logged in:        0
  Usage of /:   19.1% of 9.63GB   IP address for enp0s3:  10.0.2.15
  Memory usage: 20%               IP address for enp0s8:  10.0.200.50
  Swap usage:   0%                IP address for docker0: 172.17.0.1
  Processes:    109


  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

121 packages can be updated.
53 updates are security updates.


vagrant@nomad-server:~$
```

Situados dentro del equipo servidor virtual con **Vagrant,** vamos a comprobar que se ha instalado correctamente **Docker, Nomad Hashicorp y Consul,** en el equipo servidor virtual, junto con la versión instalada de cada herramienta descrita. Para ello, ejecutamos las siguientes secuencias de comandos a mostrarse:

- **NOMAD HASHICORP:**

```
vagrant@nomad-server:~$ nomad --version
Nomad v0.9.1 (4b2bdbd9ab68a27b10c2ee781cceaaf62e114399)
vagrant@nomad-server:~$
```

- **CONSUL:**

```
vagrant@nomad-server:~$ consul --version
Consul v1.5.1
Protocol 2 spoken by default, understands 2 to 3 (agent will automatically use protocol >2 when speaking to compatible agents)
vagrant@nomad-server:~$
```

- **DOCKER:**

```
vagrant@nomad-server:~$ docker --version
Docker version 18.09.6, build 481bc77
vagrant@nomad-server:~$
```

Como se puede observar, las herramientas **Docker, Nomad Hashicorp y Consul** se han instalado correctamente en el equipo servidor virtual.

10. Iniciar Agente Nomad Hashicorp.

Cuando hayamos creado el equipo servidor virtual, con las herramientas necesarias para la orquestación de contenedores **Docker** con **Nomad Hashicorp,** vamos a iniciar el agente **Nomad Hashicorp** en el equipo servidor virtual, para poder realizar la orquestación de contenedores **Docker. Para ello,** ejecutamos el siguiente comando, para poder iniciar el agente **Nomad Hashicorp:**

```
vagrant@nomad-server:~$ sudo nomad agent -dev
```

Iniciado el agente **Nomad Hashicorp** en el equipo servidor virtual, vamos a abrir otra ventana o **Terminal** del sistema operativo de la máquina anfitriona, y vamos a conectarnos de nuevo al equipo servidor virtual, y ya conectados a dicho equipo servidor virtual **(desde otra ventana o consola, ya que se trata de un sólo nodo),** vamos a comprobar el estado del agente o nodo de **Nomad Hashicorp** del equipo servidor virtual, mediante la ejecución del siguiente comando:

```
vagrant@nomad-server:~$ nomad node status
ID        DC   Name          Class   Drain  Eligibility  Status
467b7569  dc1  nomad-server  <none>  false  eligible     ready
vagrant@nomad-server:~$
```

Podemos ver el estado del agente de **Nomad Hashicorp,** añadiendo el número de **Allocs** ejecutándose con **Nomad Hashicorp,** ejecutamos la siguiente instrucción:

```
vagrant@nomad-server:~$ nomad node status -allocs
ID          DC    Name             Class    Drain   Eligibility   Status
Running Allocs
467b7569  dc1  nomad-server  <none>  false  eligible     ready   0
vagrant@nomad-server:~$
```

Con ello, podemos ver que el agente de **Nomad Hashicorp** está corriendo en el equipo servidor virtual, debido a que iniciamos el agente de **Nomad Hashicorp** previamente.

Nota: Lo más recomendable utilizar varios nodos o equipos virtuales, para obtener una mejor experiencia de la herramienta. Pero en éste caso, vamos a utilizar sólo un nodo como prueba.

Si quisiéramos ver el estado del nodo en ejecución de **Nomad Hashicorp,** de forma detallada **(versión corta),** tenemos que ejecutar el siguiente comando:

```
vagrant@nomad-server:~$ nomad node status -short 467b7569
ID          = 467b7569
Name        = nomad-server
Class       = <none>
DC          = dc1
Drain       = false
Eligibility = eligible
Status      = ready
Drivers     = docker,exec,raw_exec

Allocations
No allocations placed
vagrant@nomad-server:~$
```

Nota: Para poder ver el estado del nodo de forma un poco más detallada, ejecutamos el mismo comando, pero quitando la opción "-short", y así nos muestra el estado del nodo de Nomad Hashicorp de forma completa.

Si quisiéramos ver el estado del agente de los equipos servidores virtuales **Nomad Hashicorp,** que actúan como nodo cliente, tenemos que ejecutar el siguiente comando a reflejarse:

```
vagrant@nomad-server:~$ nomad node status -self
```

Nota: Como sólo tenemos configurado un sólo agente en el mismo nodo, el resultado que no saldrá, será la información del equipo servidor virtual que hemos creado, ya que no hay nodos clientes para ésta situación (que es lo más recomendable).

Si quisiéramos ver el estado del agente o nodo en ejecución de **Nomad Hashicorp,** de una manera detallada y más completa aún, mostrando los recursos del agente o nodo, ejecutamos el siguiente comando:

```
vagrant@nomad-server:~$ nomad node status -stats 467b7569
```

Con ello, nos muestra el estado del nodo o agente en ejecución de **Nomad Hashicorp**, pero de forma más completa que la opción **"-self".**

Y si ya quisiéramos ver el estado del agente o nodo de **Nomad Hashicorp,** de una forma más detallada y mucho más completa aún, ejecutamos el siguiente comando:

```
vagrant@nomad-server:~$ nomad node status -verbose 467b7569
```

Con el comando reflejado, podemos mostrar toda la información detalla y completa del estado del agente o nodo en ejecución de **Nomad Hashicorp.**

Podemos ver también los miembros existentes en el nodo o agente en ejecución de **Nomad Hashicorp,** mediante el siguiente comando:

```
vagrant@nomad-server:~$ nomad server members
Name                  Address     Port  Status  Leader  Protocol  Build
Datacenter  Region
nomad-server.global  127.0.0.1  4648  alive   true    2         0.9.1
dc1         global
vagrant@nomad-server:~$
```

Si quisiéramos ver los miembros existentes en el nodo o agente en ejecución de **Nomad Hashicorp,** pero de forma más detallada, lo haremos con la siguiente instrucción:

```
vagrant@nomad-server:~$ nomad server members -detailed
Name                  Address    Port  Tags
nomad-server.global                    127.0.0.1                     4648
raft_vsn=2,rpc_addr=127.0.0.1,port=4647,dc=dc1,vsn=1,mvn=1,build=0.9.1
,region=global,bootstrap=1,id=a60d736b-ca75-935f-0042-
321a43df4de7,role=nomad
vagrant@nomad-server:~$
```

Para poder parar el agente o nodo de **Nomad Hashicorp** en el equipo servidor virtual, basta con presionar **“Ctrl + Ç”** en nuestro teclado, y el agente de **Nomad Hashicorp** se detendrá correctamente en el equipo servidor virtual.

11. Crear Un Job Con Nomad Hahicorp.

En éste apartado, vamos a explicar como generar o crear un **Job** o trabajo en **Nomad Hashicorp,** para poder realizar la orquestación de contenedores **Docker** con **Nomad Hashicorp.**

Los **Jobs** se crean, para poder indicar a la herramienta de **Nomad Hashicorp,** las herramientas o tareas que tiene que ejecutar o las que debería de ejecutar.

Para ello, vamos a crear un fichero de Job con **Nomad Hashicorp,** ejecutando el siguiente comando:

```
vagrant@nomad-server:~$ nomad job init
Example job file written to example.nomad
vagrant@nomad-server:~$
```

Comprobamos que el fichero de **Job** de **Nomad Hashicorp** se ha creado correctamente, mediante la siguiente instrucción:

```
vagrant@nomad-server:~$ ls
example.nomad
vagrant@nomad-server:~$
```

A continuación, vamos a ejecutar o iniciar el **Job** de **Nomad Hashicorp** creada anteriormente, ejecutando el siguiente comando:

```
vagrant@nomad-server:~$ nomad job run example.nomad
==> Monitoring evaluation "367fa7a3"
    Evaluation triggered by job "example"
    Allocation "f746083a" created: node "467b7569", group "cache"
    Evaluation within deployment: "26572ace"
    Evaluation status changed: "pending" -> "complete"
==> Evaluation "367fa7a3" finished with status "complete"
vagrant@nomad-server:~$
```

Vamos a comprobar que el **Job** creado, se encuentra en ejecución en el equipo servidor virtual con **Nomad Hashicorp,** mostrando los detalles del **Job** en ejecución, mediante el siguiente comando:

```
vagrant@nomad-server:~$ nomad status example
ID            = example
Name          = example
Submit Date   = 2019-05-27T12:37:02Z
Type          = service
Priority      = 50
Datacenters   = dc1
Status        = running
Periodic      = false
Parameterized = false
```

```
Summary
Task Group  Queued  Starting  Running  Failed  Complete  Lost
cache       0       0         1        0       0         0

Latest Deployment
ID          = 26572ace
Status      = successful
Description = Deployment completed successfully

Deployed
Task Group  Desired  Placed  Healthy  Unhealthy  Progress Deadline
cache       1        1       1        0          2019-05-27T12:47:38Z

Allocations
ID          Node ID    Task Group   Version   Desired   Status    Created
Modified
f746083a   467b7569   cache        0         run       running   1m2s ago
27s ago
vagrant@nomad-server:~$
```

Como se puede observar, el **Job** que hemos creado anteriormente, se encuentra en ejecución en el nodo de **Nomad Hashicorp,** refiriéndose al equipo servidor virtual.

También podremos saber si el **Job** creado anteriormente, se encuentra en ejecución en el nodo de **Nomad Hashicorp** en el equipo servidor virtual, mediante el **"ID"** de la asignación del **Job** creado anteriormente, a través de la siguiente instrucción:

```
vagrant@nomad-server:~$ nomad alloc status d71b1c11
```

Si quisiéramos saber si el **Job** creado en el nodo de **Nomad Hashicorp**, se encuentra en ejecución en el equipo servidor virtual, pero de una forma un poco más acortada, ejecutaremos el siguiente comando:

```
vagrant@nomad-server:~$ nomad alloc status -short d71b1c11
```

Podremos mostrar también información adicional del **Job** credo, mostrando el estado de las **métricas** del Job creado, en el nodo de **Nomad Hashicorp** del equipo servidor virtual, mediante la siguiente instrucción:

```
vagrant@nomad-server:~$ nomad alloc status -verbose d71b1c11
```

Podremos visualizar los logs del **Job** creado en el nodo de **Nomad Hashicorp,** desde el equipo servidor virtual, a través del **"ID"** del **Job** creado. Para ello, vamos a ejecutar el siguiente comando:

```
vagrant@nomad-server:~$ nomad alloc logs d71b1c11 redis
```

Para poder saber si los logs del **Job** creado con **Nomad Hashicorp**, presenta tener errores en su creación o conexión de dicho **Job** creado, ejecutamos la siguiente siguiente instrucción:

```
vagrant@nomad-server:~$ nomad alloc logs -stderr d71b1c11 redis
```

Otra forma para poder visualizar los logs del **Job** creado con **Nomad Hashicorp**, es mediante el siguiente comando a reflejarse:

```
vagrant@nomad-server:~$ nomad alloc logs -job example
```

Si queremos visualizar las últimas líneas de los logs del **Job** creado con **Nomad Hashicorp** en el equipo servidor virtual, podremos hacerlo mediante la siguiente instrucción:

```
vagrant@nomad-server:~$ nomad alloc logs -tail -n 2 d71b1c11 redis
_hugepage/enabled' as root, and add it to your /etc/rc.local in order
to retain the setting after a reboot. Redis must be restarted after
THP is disabled.
1:M 09 Jun 11:30:14.901 * The server is now ready to accept
connections on port 6379
vagrant@nomad-server:~$
```

Como se observa en la ejecución del comando, nos informa que el servidor está preparado para poder aceptar conexiones desde el puerto **"6379"**, por lo que el **Job** creado, se está ejecutando correctamente, en el nodo de **Nomad Hashicorp** del equipo servidor virtual.

Si quisiéramos visualiza los logs del **Job** creado en el nodo de **Nomad Hashicorp "en tiempo real"**, vamos a ejecutar el siguiente comando:

```
vagrant@nomad-server:~$ nomad alloc logs -tail -f -n 3 d71b1c11 redis
```

12. Modificar Un Job Con Nomad Hahicorp.

Cuando creamos un **Job** en el nodo de **Nomad Hashicorp** del equipo servidor virtual, nos creará un fichero de configuración, donde podemos editarlo, para poder actualizar la aplicación a desplegar, el número de contenedores a ejecutar o cambiar el número grupo de tareas a ejecutar en el nodo de **Nomad Hashicorp.**

Para ésta situación, he editado el grupo de tareas a ejecutar en el nodo de **Nomad Hashicorp,** cambiando a tres grupos de tareas a ejecutar con **Nomad Hashicorp (por defecto un grupo de trabajo a ejecutar).** Para ello, he editado el fichero del **Job** creado con **Nomad Hashicorp,** a través de la siguiente instrucción:

```
vagrant@nomad-server:~$ sudo nano example.nomad
```

Situados dentro del fichero descrito anteriormente, vamos a editar el siguiente parámetro, para poder cambiar el grupo de trabajos del **Job** a ejecutar con **Nomad Hashicorp:**

```
# The "count" parameter specifies the number of the task groups that
should
# be running under this group. This value must be non-negative and
defaults
# to 1.
count = 3
```

Guardamos los cambios efectuados y salimos de dicho fichero de configuración del **Job** creado en el nodo de **Nomad Hashicorp.**

Cuando hayamos editado el parámetro anteriormente descrito, en el fichero de configuración del Job creado en el nodo de **Nomad Hashicorp,** tenemos que actualizar el plan del **Job** creado, a través de la siguiente instrucción:

```
vagrant@nomad-server:~$ nomad job plan example.nomad
+/- Job: "example"
+/- Task Group: "cache" (2 create, 1 in-place update)
  +/- Count: "1" => "3" (forces create)
      Task: "redis"

Scheduler dry-run:
- All tasks successfully allocated.

Job Modify Index: 13
To submit the job with version verification run:

nomad job run -check-index 13 example.nomad

When running the job with the check-index flag, the job will only be run if the
server side version matches the job modify index returned. If the index has
changed, another user has modified the job and the plan's results are
potentially invalid.
vagrant@nomad-server:~$
```

Como podemos observar, se han creado dos nuevas instancias en el **Job** del nodo **Nomad Hashicorp** del equipo servidor virtual, debido al parámetro editado en el fichero de configuración del **Job** creado.

Si queremos comprobar que la configuración del **Job,** no ha sido editado desde que se creó el plan del **Job** en el nodo de **Nomad Hashicorp,** podemos ejecutar el siguiente comando:

```
vagrant@nomad-server:~$ nomad job run -check-index 13 example.nomad
==> Monitoring evaluation "5344e54c"
    Evaluation triggered by job "example"
    Allocation "44f6f151" created: node "1814f000", group "cache"
    Allocation "a4443b58" created: node "1814f000", group "cache"
    Allocation "d71b1c11" modified: node "1814f000", group "cache"
    Evaluation within deployment: "74c75290"
    Evaluation status changed: "pending" -> "complete"
==> Evaluation "5344e54c" finished with status "complete"
vagrant@nomad-server:~$
```

Como hemos editado la configuración del Job creado, en el nodo de **Nomad Hashicorp** del equipo servidor virtual, **Nomad Hashicorp** ha creado dos asignaciones más, haciendo que hayan creadas tres asignaciones o grupos de trabajo, en vez de una asignación o grupo de trabajo definido por defecto.

Para éste caso, la aplicación que voy a desplegar, es una aplicación llamada **“Redis”.** Para ello, he editado el fichero de configuración del **Job** creado, en el nodo de **Nomad Hashicorp** del equipo servidor virtual, y he cambiado la versión a desplegar u orquestar con contenedores **Docker,** la aplicación **Redis** con **Nomad Hashicorp.**

```
vagrant@nomad-server:~$ sudo nano example.nomad
```

```
task "nginx" {
driver = "docker"

config {
      image = "nginx:latest"
      port_map {
      db = 80
      }
}
```

Guardamos los cambios efectuados y salimos de dicho fichero.

Volvemos a actualizar el plan del **Job** creado en el nodo de **Nomad Hashicorp,** ejecutando el siguiente comando:

```
vagrant@nomad-server:~$ nomad job plan example.nomad
+/- Job: "example"
+/- Task Group: "cache" (1 create/destroy update, 2 ignore)
  +/- Task: "redis" (forces create/destroy update)
    +/- Config {
      +/- image:           "redis:3.2" => "redis:latest"
          port_map[0][db]: "6379"
        }

Scheduler dry-run:
- All tasks successfully allocated.

Job Modify Index: 94
To submit the job with version verification run:

vagrant@nomad-server:~$ nomad job run -check-index 94 example.nomad

When running the job with the check-index flag, the job will only be run if the
server side version matches the job modify index returned. If the index has
changed, another user has modified the job and the plan's results are
potentially invalid.
vagrant@nomad-server:~$
```

Como se puede ver, el parámetro de la versión de la aplicación a desplegar, ha cambiado en el plan del **Job** creado, en el nodo de **Nomad Hashicorp** en el equipo servidor virtual.

Volvemos a iniciar el **Job** creado en el nodo de **Nomad Hashicorp,** pero ya con las modificaciones realizadas en los pasos anteriores, a través de la siguiente instrucción:

```
vagrant@nomad-server:~$ nomad job run example.nomad
==> Monitoring evaluation "1c70e5f1"
    Evaluation triggered by job "example"
    Evaluation within deployment: "90cb8485"
    Allocation "22be0abb" created: node "7c9029b0", group "cache"
    Allocation "3d103db5" created: node "7c9029b0", group "cache"
    Allocation "9114e2e4" created: node "7c9029b0", group "cache"
    Evaluation status changed: "pending" -> "complete"
==> Evaluation "1c70e5f1" finished with status "complete"
vagrant@nomad-server:~$
```

Si quisiéramos para el **Job** en ejecución en el nodo de **Nomad Hashicorp,** en el equipo servidor virtual, lo podemos hacer mediante el siguiente comando:

```
vagrant@nomad-server:~$ nomad job stop example
```

13. Crear Primer Clúster Con Nomad Hahicorp.

En éste apartado, se explicará como crear y configurar nuestro primer clúster, a través de **Nomad Hashicorp**. Para ello, los pasos que tenemos que seguir para la creación y configuración de nuestro primer clúster con **Nomad Hashicorp**, son los siguientes a mostrarse a continuación.

Primero, vamos a crear un fichero con extensión **“.hcl”**, donde seguidamente, será modificado, con la configuración del clúster a crear, configurar y ejecutar en el nodo de **Nomad Hashicorp.** En mi caso, he creado un fichero llamado **“server.hcl”**, y acto seguido, he editado el fichero de configuración creado, con el siguiente contenido a reflejarse:

```
vagrant@nomad-server:~$ sudo nano server.hcl

# Increase log verbosity
log_level = "DEBUG"

# Setup data dir
data_dir = "/tmp/server1"

# Give the agent a unique name. Defaults to hostname
name = "server1"

# Enable the server
server {
  enabled = true

  # Self-elect, should be 3 or 5 for production
  bootstrap_expect = 1
}
```

Guardamos los cambios efectuados y salimos de dicho fichero de configuración.

Comprobamos que el fichero de configuración del **clúster** a crear con **Nomad Hashicorp** anteriormente descrito, se ha creado correctamente en el equipo servidor virtual, mediante la siguiente instrucción:

```
vagrant@nomad-server:~$ ls -lrt
total 20
-rw-rw---- 1 vagrant vagrant 15673 Jun  9 13:19 example.nomad
-rw-r--r-- 1 root    root      281 Jun  9 13:45 server.hcl
vagrant@nomad-server:~$
```

Como se puede observar, el fichero de configuración del clúster de **Nomad Hashicorp** se ha creado correctamente en el equipo servidor virtual.

A continuación, vamos a proceder a iniciar el agente del clúster de **Nomad Hashicorp** en el equipo servidor, a través del fichero de configuración del clúster creada anteriormente. Para ello, ejecutamos el siguiente comando:

```
vagrant@nomad-server:~$ nomad agent -config server.hcl
```

Con ello, hemos iniciado el agente de clúster de **Nomad Hashicorp** creada anteriormente, en el equipo servidor virtual.

Seguidamente, vamos a crear los clientes para el clúster creado anteriormente de **Nomad Hashicorp.** En mi caso, voy a crear dos clientes para dicho clúster, que serán los clientes **“client1”** y **“client2”.** Para ello, crearemos dos ficheros de configuración **“.hcl”,** que serán los ficheros **“client1.hcl”** y **“client2.hcl”.**

Aquí se mostrará la creación de dichos ficheros de configuración, con la correspondiente configuración que tendrá cada uno de los ficheros de configuración de los clientes a crear, para el clúster de **Nomad Hashicorp.**

```
vagrant@nomad-server:~$ sudo nano client1.hcl
```

```
# Increase log verbosity
log_level = "DEBUG"

# Setup data dir
data_dir = "/tmp/client1"

# Give the agent a unique name. Defaults to hostname
name = "client1"

# Enable the client
client {
  enabled = true
```

```
  # For demo assume we are talking to server1. For production,
  # this should be like "nomad.service.consul:4647" and a system
  # like Consul used for service discovery.
  # servers = ["127.0.0.1:4647"]
  servers = ["0.0.0.0:4647"]
}

# Modify our port to avoid a collision with server1
ports {
  http = 5656
}

vagrant@nomad-server:~$ sudo nano client2.hcl

# Increase log verbosity
log_level = "DEBUG"

# Setup data dir
data_dir = "/tmp/client2"

# Give the agent a unique name. Defaults to hostname
name = "client2"

# Enable the client
client {
  enabled = true

  # For demo assume we are talking to server1. For production,
  # this should be like "nomad.service.consul:4647" and a system
  # like Consul used for service discovery.
  # servers = ["127.0.0.1:4647"]
  servers = ["0.0.0.0:4647"]
}
```

```
# Modify our port to avoid a collision with server1 and client1
ports {
  http = 5657
}
```

Decir que tenemos que guardar los cambios efectuados, en cada uno de los ficheros de configuración de los clientes a crear para el clúster de **Nomad Hashicorp.**

En el siguiente paso, vamos a iniciar los agentes de cada cliente, mediante los ficheros de configuración previamente creado de cada cliente. Para ello, ejecutamos los siguientes comandos:

```
vagrant@nomad-server:~$ nomad agent -config client1.hcl
vagrant@nomad-server:~$ nomad agent -config client2.hcl
```

Con los comandos anteriormente descritos y ejecutados, ya habremos iniciado el agente de los dos clientes para el clúster creado de **Nomad Hashicorp**, en el nodo del equipo servidor virtual.

Vamos a comprobar el estado actual del nodo de **Nomad Hashicorp** en ejecución, en el equipo servidor virtual, a través de la siguiente instrucción:

```
vagrant@nomad-server:~$ nomad node status
ID        DC   Name     Class   Drain  Eligibility  Status
c41eae53  dc1  client2  <none>  false  eligible     ready
ede12777  dc1  client1  <none>  false  eligible     ready
vagrant@nomad-server:~$
```

Como se observa, los dos clientes creados, se encuentran ejecutándose en el nodo de **Nomad Hashicorp** del equipo servidor virtual.

14. Enviar Job Creado En Nomad Hashicorp.

Cuando tengamos creado un clúster en **Nomad Hashicorp**, con varios clientes ejecutándose en el nodo de **Nomad Hashicorp**, donde en éste caso, será el equipo servidor virtual donde estamos trabajando, vamos a enviar un **Job** que hemos creado previamente, con la correspondiente configuración adaptada a nuestro entorno de trabajo. Para ello, vamos a ejecutar la siguiente instrucción:

```
vagrant@nomad-server:~$ nomad job run example.nomad
==> Monitoring evaluation "df6bb73a"
    Evaluation triggered by job "example"
    Evaluation within deployment: "c013e898"
    Allocation "077b1629" created: node "ede12777", group "cache"
    Allocation "88b1d288" created: node "c41eae53", group "cache"
    Allocation "fb48d99a" created: node "ede12777", group "cache"
    Evaluation status changed: "pending" -> "complete"
==> Evaluation "df6bb73a" finished with status "complete"
vagrant@nomad-server:~$
```

Como se puede observar, hemos iniciado el **Job** creado previamente, en el nodo de **Nomad Hashicorp** del equipo servidor virtual.

A continuación, vamos a comprobar el estado del **Job** que hemos iniciado con anterioridad en **Nomad Hashicorp**, mediante el siguiente comando:

```
vagrant@nomad-server:~$ nomad status example
ID            = example
Name          = example
Submit Date   = 2019-06-15T10:50:52Z
Type          = service
Priority      = 50
Datacenters   = dc1
Status        = running
Periodic      = false
Parameterized = false
```

```
Summary
Task Group  Queued  Starting  Running  Failed  Complete  Lost
cache       0       3         0        0       0         0

Latest Deployment
ID          = c013e898
Status      = running
Description = Deployment is running

Deployed
Task Group  Desired  Placed  Healthy  Unhealthy  Progress Deadline
cache       3        3       0        0          2019-06-15T11:00:52Z

Allocations
ID          Node ID    Task Group   Version   Desired   Status    Created
Modified
077b1629   ede12777   cache        0         run       pending   41s ago
41s ago
88b1d288   c41eae53   cache        0         run       pending   41s ago
41s ago
fb48d99a   ede12777   cache        0         run       pending   41s ago
41s ago
vagrant@nomad-server:~$
```

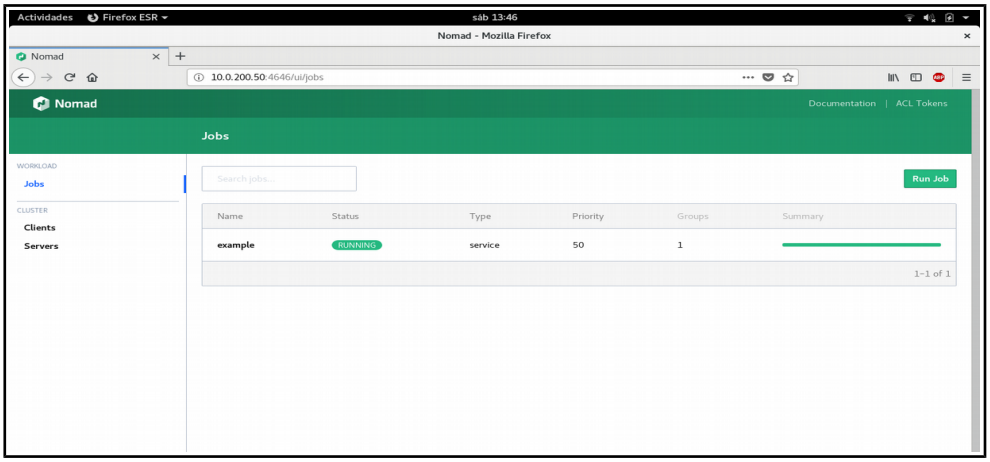
Como se puede observar, la configuración establecida en el **Job** creado y configurado con **Nomad Hashicorp**, es la ejecución de tres trabajos, donde dos trabajos o **Job** se están ejecutando en el primer cliente, y el tercer trabajo o **Job**, se está ejecutando en el segundo cliente.

15. Acceder A Web-UI De Nomad Hashicorp.

Llegados ya a éste punto, vamos a acceder a la consola gráfica **"Web-UI"** de **Nomad Hashicorp,** para poder llevar la gestión de los **Jobs** creados, configurados y ejecutándose, en el nodo de **Nomad Hashicorp** del equipo servidor virtual.

La consola **“Web-UI”** de **Nomad Hashicorp,** nos va a servir para poder inspeccionar o ver los clústers, jobs, despliegues, evaluaciones, tareas en grupo, asignaciones, logs, clientes y servidores existentes en el nodo de **Nomad Hashicorp,** para poder realizar la monitorización de contenedores **Docker,** donde en ellas, tienen implantadas una o varias aplicaciones webs o despliegues de aplicaciones.

Para poder acceder a la consola gráfica **“Web-UI”** de **Nomad Hashicorp** del equipo servidor virtual donde se está ejecutando, vamos a acceder a nuestro navegador web que utilizamos habitualmente, y en la barra de navegación o **URL**, accedemos a la dirección web http://10.0.200.50:4646 .



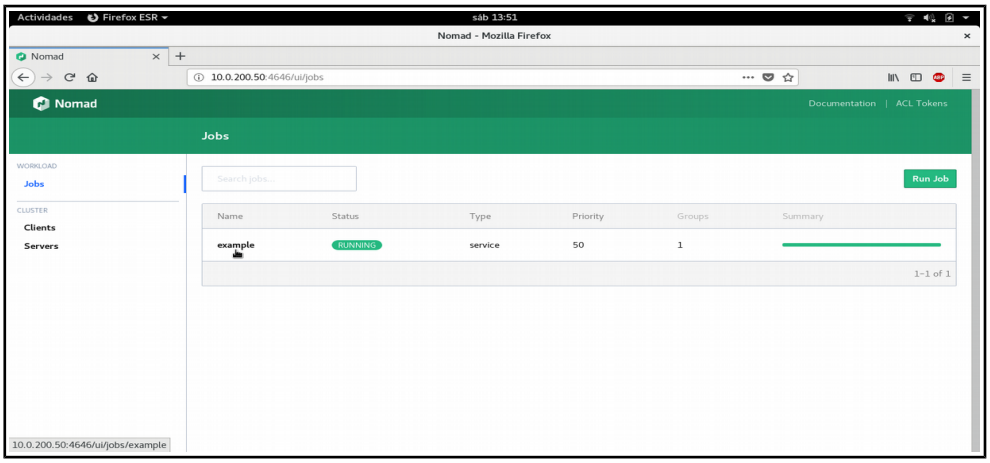
Como resultado, obtenemos acceso a la consola gráfica **“Web-UI”** de **Nomad Hashicorp**, mostrando los **Jobs** que se están ejecutando actualmente, en dicho nodo de **Nomad Hashicorp.**

16. Web-UI De Nomad Hashicorp.

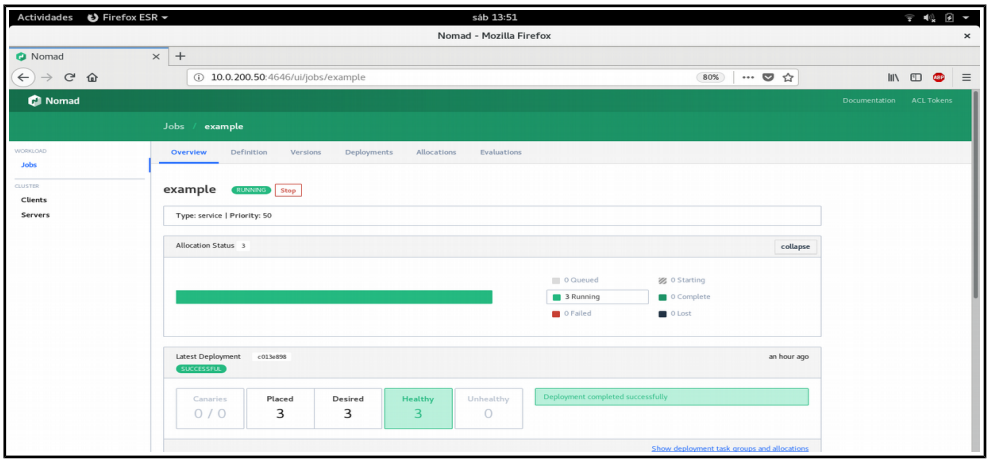
Cuando hayamos accedido a la consola gráfica **“Web-UI”** del nodo de **Nomad Hashicorp** en ejecución, en el equipo servidor virtual donde estamos trabajando, poder ir viendo las diferentes opciones que podemos ir gestionando y monitorizando en dicha consola gráfica.

En el acceso a la consola gráfica **“Web-UI”** de **Nomad Hashicorp**, podemos ver que nos deja situados en la pestaña **“Jobs”**, donde se mostrarán todos los trabajos ejecutándose en el nodo de **Nomad Hashicorp** actualmente.

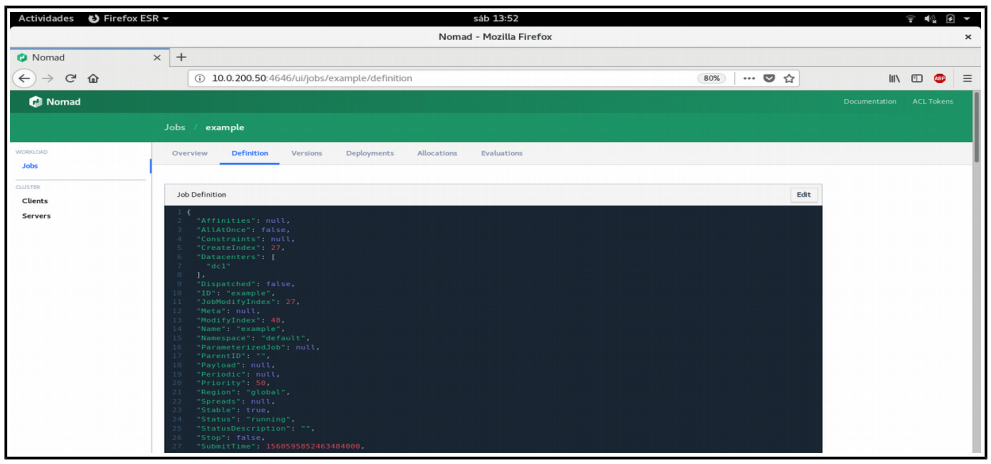
Tenemos la opción de poder pinchar en el **Job** que queramos, y que esté ejecutándose en el nodo de **Nomad Hashicorp**, para poder mostrarnos más información sobre dicho **Job** o trabajo ejecutándose.



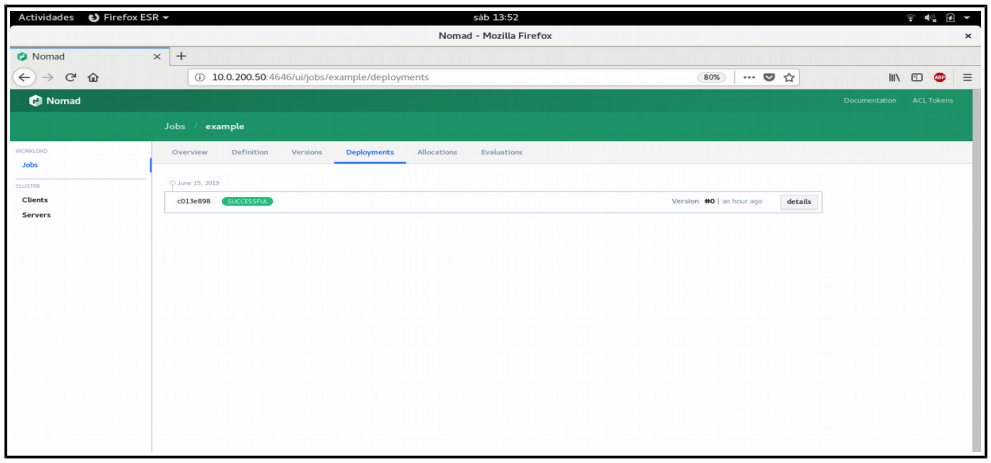
Situados dentro del Job donde queremos inspeccionar un poco más a fondo, podemos ver que nos deja situados en la pestaña **“Overview”**, donde nos muestra un resumen del estado del **Job** que se está ejecutando en el nodo de **Nomad Hashicorp.**



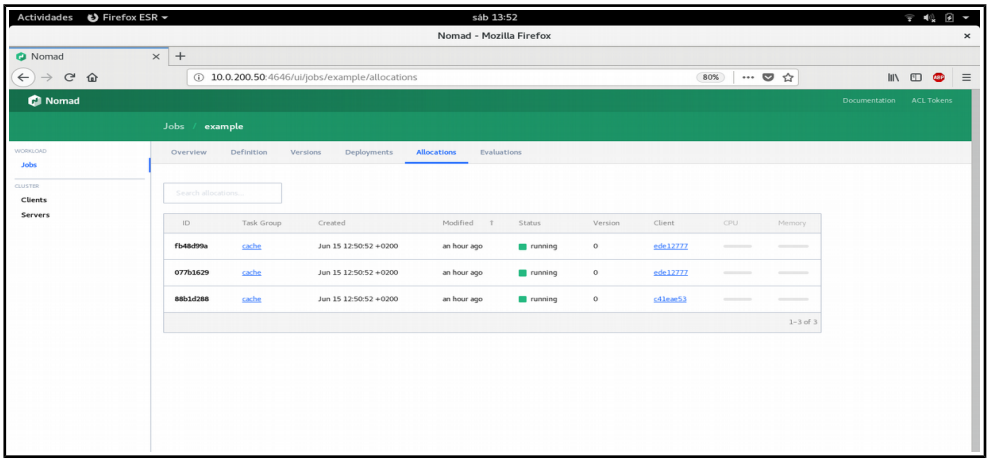
Si nos dirigimos a la pestaña **“Definition”**, nos mostrará toda la definición del **Job** o trabajo ejecutándose en el nodo de **Nomad Hashicorp.**



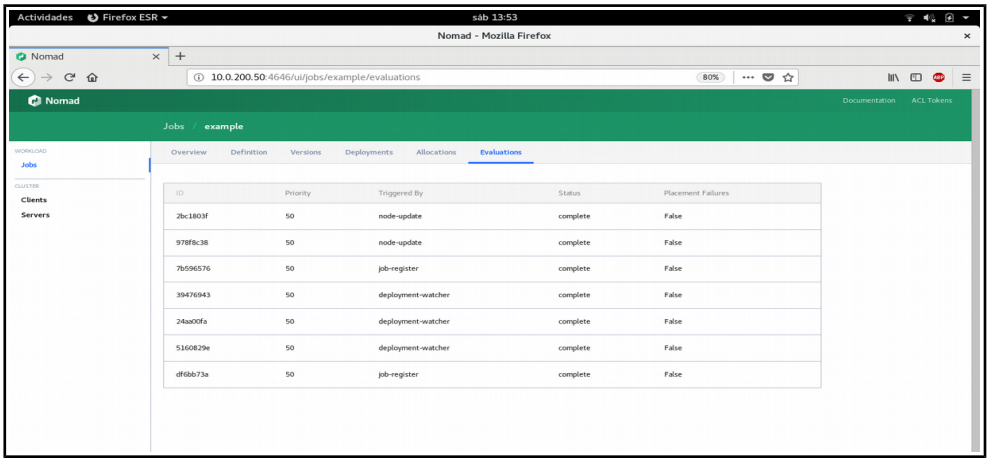
En la pestaña **“Deployments”,** nos mostrará los despliegues realizados en el **Job** seleccionado y ejecutándose en el nodo de **Nomad Hashicorp.**



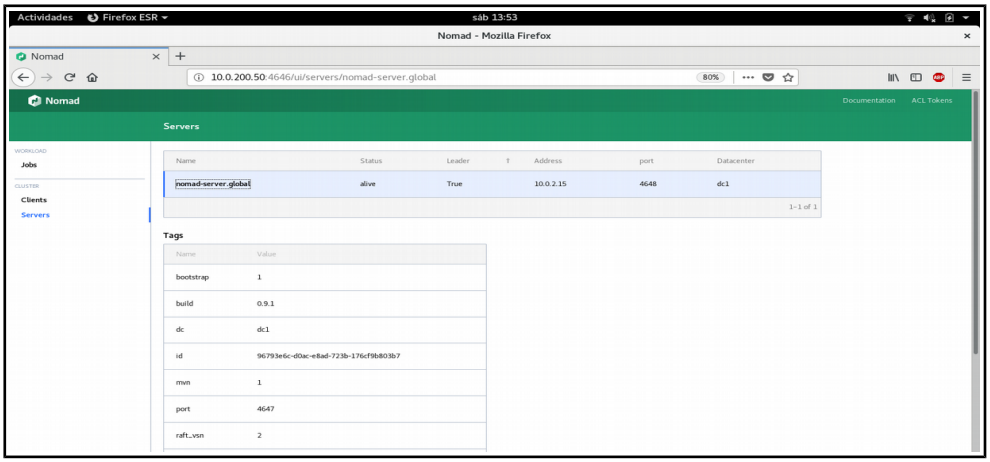
Si nos dirigimos a la pestaña **“Allocations”**, nos mostrará las asignaciones configuradas y ejecutadas en en Job seleccionado de **Nomad Hashicorp**, donde en éste caso, se está ejecutando tres asignaciones, que son las asignaciones que se configuraron previamente, en el fichero de configuración del Job a ejecutar, en el nodo de **Nomad Hashicorp.**



En la pestaña **“Evaluations”**, nos mostrará las evaluaciones del **Job** ejecutándose, en el nodo de **Nomad Hashicorp** en el equipo servidor virtual.

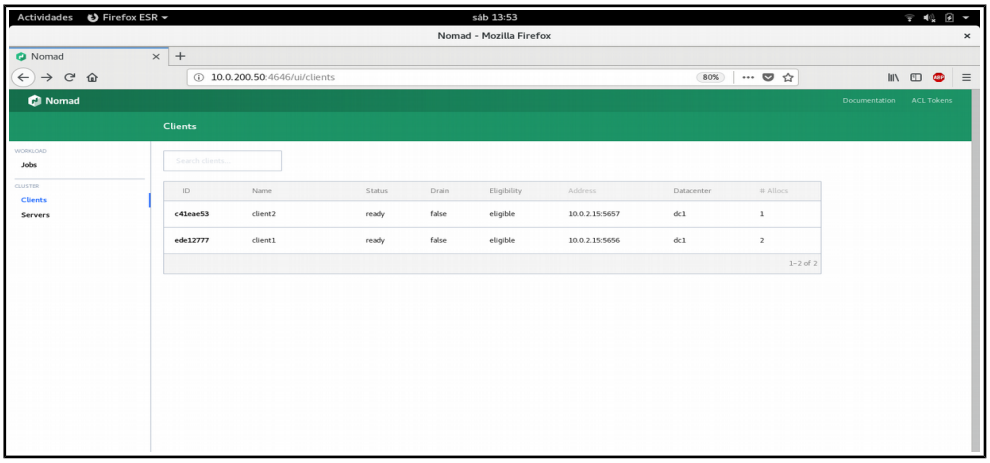


Mediante la consola **"Web-UI"** de **Nomad Hashicorp**, podemos ver la información de los equipos Servidores y equipos Clientes configurados, en el nodo de **Nomad Hashicorp** ejecutándose en el equipo servidor virtual.



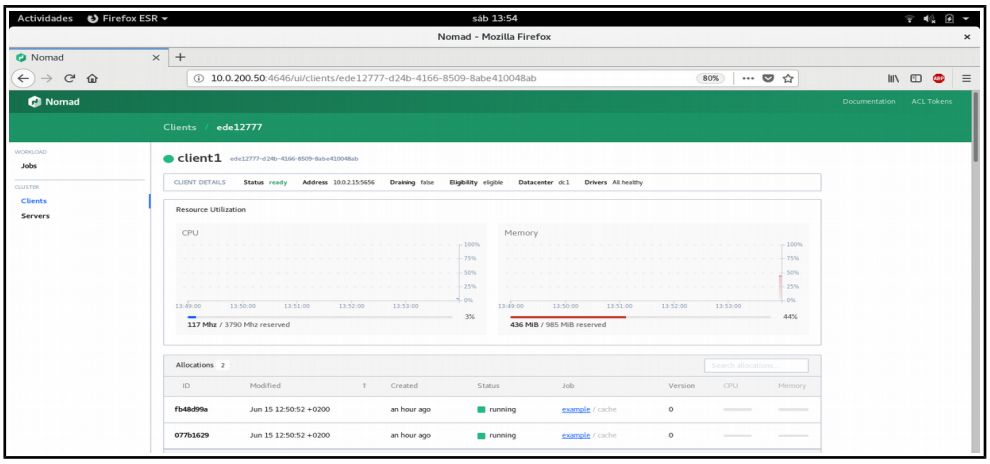
En nuestro caso, sólo tenemos configurado un equipo Servidor para **Nomad Hashicorp,** ya sea para la gestión de **Nomad Hashicorp,** como para la gestión de varios **Jobs** creados, en uno o varios nodos de **Nomad Hashicorp.**

Si nos dirigimos a la pestaña **"Clients",** podemos ver los clientes disponibles para **Nomad Hashicorp**, donde en nuestro caso, tenemos configurados dos equipos **Clientes** para **Nomad Hashicorp.**

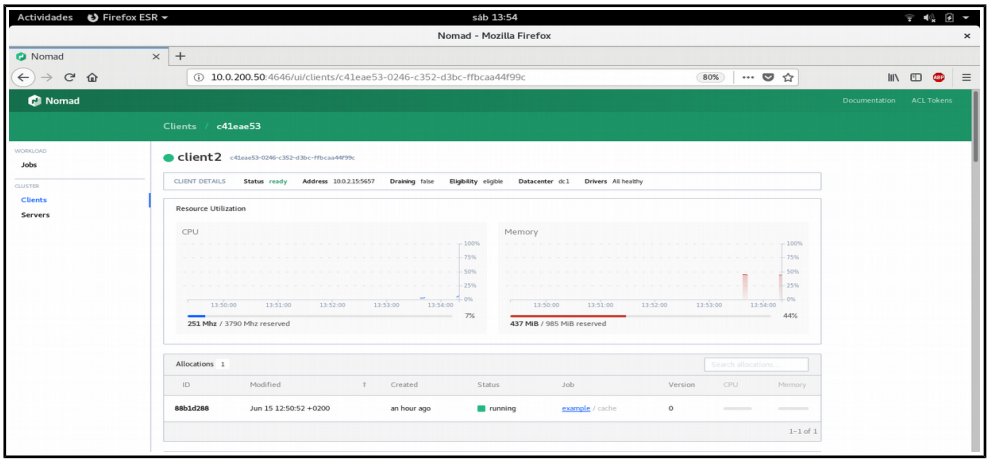


Si pinchamos encima de cada uno de los equipos, ya sean Servidores o Clientes, nos mostrará la información de cada equipo Servidor o Cliente creado y configurado, para el nodo de **Nomad Hashicorp** en ejecución.

- **Equipo Cliente 1.**



- **Equipo Cliente 2.**



En cada uno de los equipos Clientes previamente mostrado, se puede ver un breve resumen de los atributos de cada equipo, el consumo en memoria y CPU utilizada en cada equipo, las asignaciones ejecutadas, y los eventos realizados en cada uno de los equipos clientes.

Éstas son las opciones más comunes y sencillas de poder ver en la consola gráfica **"Web-UI"** de **Nomad Hashicorp**. Podemos inspeccionar y monitorizar también con más profundidad, las asignaciones ejecutadas en cada equipo, las evaluaciones, definiciones de **Jobs** ejecutados en cada equipo, y otras opciones más derivadas a los **Jobs** ejecutados con **Nomad Hashicorp.**

17. Comprobar Orquestación Docker Con Nomad Hashicorp.

Por último, vamos a comprobar la orquestación de contenedores **Docker**, se ha realizado correctamente, para la aplicación desplegada en el nodo de **Nomad Hashicorp,** donde en éste caso, la aplicación desplegada ha sido **"Redis".**

Para ello, vamos a comprobar los números de contenedores **Docker** creado, mediante el Job ejecutado en el nodo de **Nomad Hashicorp** del equipo servidor virtual. En éste caso, vamos a ejecutar el siguiente comando:

```
vagrant@nomad-server:~$ docker ps
CONTAINER ID                IMAGE                                COMMAND
CREATED                         STATUS                                PORTS
NAMES
ca2d20b7b7db          redis:latest          "docker-entrypoint.s…"    5
minutes ago          Up 5 minutes           10.0.2.15:28007->6379/tcp,
10.0.2.15:28007->6379/udp   redis-6cd72cbe-42d0-4b60-26a5-d7ce3c14e707
16f242b76462          redis:latest          "docker-entrypoint.s…"    5
minutes ago          Up 5 minutes           10.0.2.15:22647->6379/tcp,
10.0.2.15:22647->6379/udp   redis-b3f6548d-a1ff-db12-fa8f-6f61b8e1677f
515a33de9c0b          redis:latest          "docker-entrypoint.s…"    5
minutes ago          Up 5 minutes           10.0.2.15:31637->6379/tcp,
10.0.2.15:31637->6379/udp   redis-4a3a3bc6-407f-f239-ea2e-9a4c71b76de0
vagrant@nomad-server:~$
```

Como se puede observar, se ha creado tres contenedores **Docker**, para el despliegue de la aplicación **"Redis"**, en el nodo de **Nomad Hashicorp** del equipo servidor virtual. La cantidad de contenedores a crearse para la aplicación desplegada, se ha definido en el fichero de configuración del **Job** en ejecución.

Podemos acceder a cada uno de los contenedores **Docker** creados, en el nodo de **Nomad Hashicorp** del equipo servidor virtual. Para ello, tenemos que ejecutar las siguientes instrucciones, para poder acceder a cada contenedor creado para la aplicación desplegada.

- **Contenedor Docker ca2d20b7b7db**:

```
vagrant@nomad-server:~$ docker exec -i -t ca2d20b7b7db /bin/bash
root@ca2d20b7b7db:/data# pwd
/data
root@ca2d20b7b7db:/data# exit
exit
vagrant@nomad-server:~$
```

- **Contenedor Docker 16f242b76462**:

```
vagrant@nomad-server:~$ docker exec -i -t 16f242b76462 /bin/bash
root@16f242b76462:/data# pwd
/data
root@16f242b76462:/data# exit
exit
vagrant@nomad-server:~$
```

- **Contenedor Docker 515a33de9c0b**:

```
vagrant@nomad-server:~$ docker exec -i -t 515a33de9c0b /bin/bash
root@515a33de9c0b:/data# pwd
/data
root@515a33de9c0b:/data# exit
exit
vagrant@nomad-server:~$
```

Como se puede observar, podemos acceder a los tres contenedores **Docker** creados en el nodo de **Nomad Hashicorp**, situándonos en el directorio **"/data"** de cada contenedor.

18. Conclusión.

A lo largo que he ido realizando el proyecto de **Nomad Hashicorp,** me he dado cuenta que hay objetivos que he podido conseguir, y hay otros que no he podido conseguir hacer **(sin contar con futuros objetivos a poder realizar con la herramienta Nomad Hashicorp).**

Me ha parecido una herramienta interesante, para poder realizar la **"orquestación"** de contenedores **Docker,** con una consola gráfica **"Web-UI"** propia, para poder encargarnos de la gestión y monitorización del diferentes funcionalidades de la orquestación de contenedores **Docker** con **Nomad Hashicorp.**

Pero mi conclusión final, es que más que realizar una orquestación de contenedores **Docker** la herramienta **Nomad Hashicorp,** se encarga más de la gestión y monitorización de contenedores **Docker** de aplicaciones desplegadas, haciendo que si hubiese cualquier problema en algunos de los contenedores desplegados u orquestados, poder tratar la/s incidencia/s que estén ocasionando en una o varias aplicación/es desplegadas en uno o varios contenedores **Docker** de uno o varios nodos de **Nomad Hashicorp.** Aunque en realidad también realiza orquestación de contenedores **Docker,** de una o varias aplicaciones configuradas, para ser desplegada en uno o varios nodos de **Nomad Hashicorp.**

En definitiva, **Nomad Hashicorp** no es una buena opción para poder realizar la orquestación de muchos contenedores **Docker,** para aquellas aplicaciones que sean más pesadas de ejecutar en un equipo servidor **(para ello es mejor Kubernetes).** Sin embargo, es una buena opción para poder realizar una pequeña orquestación de contenedores **Docker,** para una o varias aplicaciones que consuman pocos recursos en un equipo servidor.

Nunca está mal poder encontrar una herramienta que te permita poder orquestar contenedores **Docker,** y llevar una gestión y monitorización de los contenedores **Docker** generados, para el despliegue de una o varias aplicaciones con la herramienta **Nomad Hashicorp.** Siempre es buscar otras opciones o variantes similares a **Kubernetes,** ya que es la herramienta que más se utiliza, para la orquestación de contenedores **Docker** en nube, y por llevar una muy buena gestión de los contenedores **Docker,** a la hora de hacer una orquestación de contenedores, para una o varias aplicaciones a desplegar.

19. Trabajo Futuro Para Nomad Hashicorp.

La herramienta **Nomad Hashicorp,** atendiendo a lo que se ha escrito en la conclusión de dicha herramienta utilizada, tiene multitud de opciones, para poder realizar la orquestación, despliegue de aplicaciones, gestión y monitorización de contenedores **Docker** desde consola **"Web-UI"** de **Nomad Hashicorp,** que nos puede permitir poder realizar una orquestación, con una amplitud un poco más extensa a lo realizado en el proyecto.

Poder implementarlo con otras herramientas como por ejemplo **Vault Hashicorp, Docker Swarm, AWS, Kubernetes** y aplicaciones derivadas para la orquestación de contenedores **Docker,** sería una opción muy buena, para poder realizar orquestación de contenedores **Docker** para aplicaciones en local y en la nube, y llevar una gestión y monitorización de los contenedores de **Docker,** para las aplicaciones desplegadas en una infraestructura con varios equipos servidores.

También se podría implementar para **Nomad Hashicorp,** la utilización de herramientas como **Terraform, Ansible, Puppet y Chef,** para poder crear diferentes escenarios de orquestación de contenedores **Docker,** a través de la herramienta **Nomad Hashicorp,** a parte de poder hacer uso de **Vagrant, VMWare, Virtualbox** u otra herramienta de virtualización, para los equipos servidores a crear para el escenario a definir, para poder realizar la orquestación de contenedores **Docker** con **Nomad Hashicorp.** Básicamente, poder realizar la configuración automática, de todas las herramientas necesarias, para que una vez creado el escenario, podamos tener aplicaciones desplegadas con **Nomad Hashicorp.**

Como última implementación para un trabajo futuro, es poder ver más a fondo, la herramienta de **Consul Hashicorp,** ya que es una muy buena herramienta para **Nomad Hashicorp,** debido a que es una malla de servicios distribuidos para conectar, asegurar y configurar servicios en cualquier plataforma de tiempo de ejecución, y en una nube pública o privada.

Éstas son las diferentes opciones o ideas que he podido idear, para poder realizar un trabajo futuro, para la aplicación de **Nomad Hashicorp.** Hay muchas más opciones a poder realizar con la herramienta de **Nomad Hashicorp,** que se pueden idear, para poder ampliar los conocimientos, acerca de la herramienta utilizada y aprendida hasta el momento, que es **Nomad Hashicorp.**

20. Referencias.

- https://www.hashicorp.com/
- https://www.nomadproject.io/
- https://www.consul.io/
- https://www.vaultproject.io/
- https://www.vagrantup.com/
- https://redis.io/
- https://www.terraform.io/
- https://aws.amazon.com/es/
- https://azure.microsoft.com/es-es/free/search/?&OCID=AID719820_SEM_OrQXEivm&lnkd=Google_Azure_Brand&dclid=CKPmrdOZ7uICFQZ00wod1AEFUQ
- https://kubernetes.io/
- https://www.ansible.com/
- https://httpd.apache.org/
- https://www.nginx.com/
- https://www.debian.org/
- https://ubuntu.com
- https://www.redhat.com/es/technologies/linux-platforms/enterprise-linux