



PROYECTO INTEGRADO

**DESPLIEGUE CON TERRAFORM
DE UN ENTORNO WORDPRESS
CON ALTA DISPONIBILIDAD Y
RECUPERACIÓN ANTE
DESASTRES EN AZURE**

POR MARÍA JESÚS ALLOZA
RODRÍGUEZ

ÍNDICE

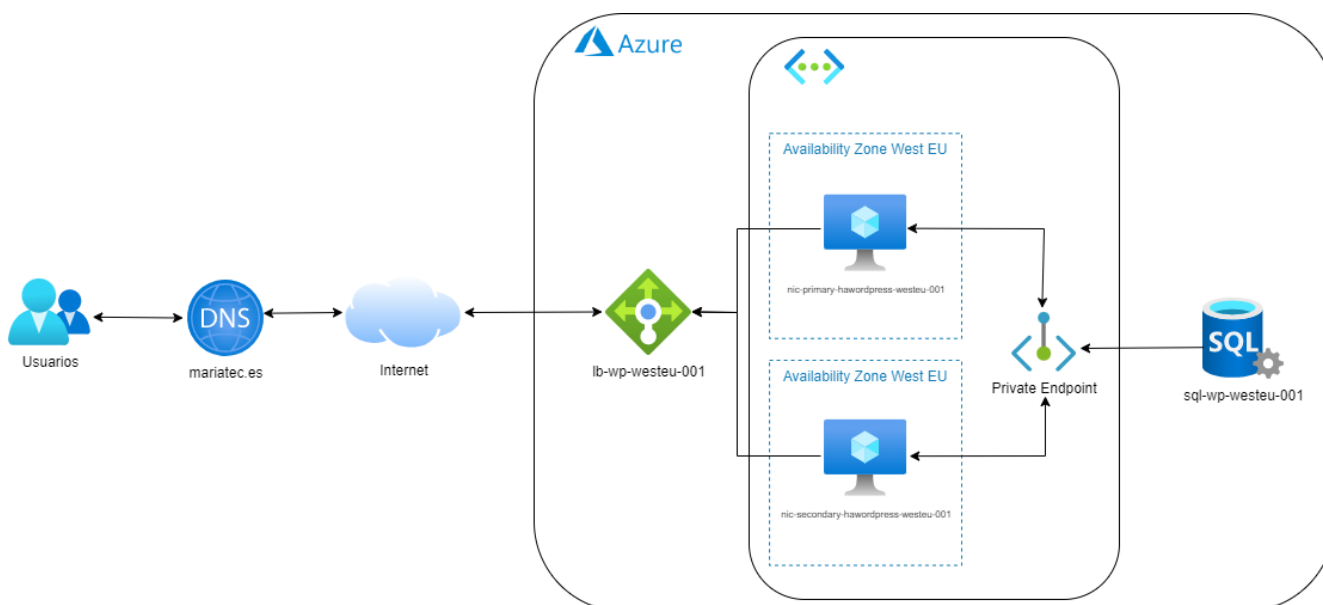
- 1. Descripción 2
 - 1. Tecnologías empleadas. 3
 - 2. Resultados esperados..... 3
- 2. ¿Qué es la nube?..... 4
 - 2.1. ¿Qué es la informática en la nube? 4
 - 2.2. Tipos de nube 4
 - 2.3. Beneficios y consideraciones de la nube 4
 - 2.4. Servicios en la nube..... 6
- 3. ¿Qué es Terraform? 7
 - 3.1. ¿Cómo funciona Terraform?..... 7
 - 3.2. Conceptos claves de Terraform 8
 - 3.3. Cómo se instala en WSL (Windows Subsystem for Linux) 9
- 4. ¿Qué es Azure?..... 11
 - 4.1. ¿Cómo funciona Azure? 11
 - 4.2. Conceptos claves de Azure 12
 - 4.3. Precios de Azure 13
 - 4.4. ¿Cómo podemos usar Azure? 14
 - 4.4.1. Azure Portal 14
 - 4.4.2. Azure CLI 15
 - 4.4.2.1. Instalación de Azure CLI en Windows..... 15
 - 4.1.1.1. Instalación de Azure CLI en MacOS 16
 - 4.4.3. A través de herramientas de IaC (Infrastructure as a Code)..... 16
- 5. Obtención de cuenta en Azure 17
- 6. Parte práctica..... 18
 - 6.1. Objetivos 18
 - 6.2. Requisitos 18
 - 6.3. Pasos a seguir..... 18
- 8. Conclusión 27
- 9. Bibliografía..... 28

1. Descripción

Con la llegada de los servicios Cloud, cada vez más empresas transfieren sus **servicios on-premises a servicios públicos o de forma híbrida**. Por ello, como dice el título del proyecto, vamos a realizar un despliegue automatizado en una nube pública, en este caso, será en **Microsoft Azure**. La infraestructura la construiremos con **Terraform**, donde declararemos todo necesario para desplegar el servicio elegido, que será **WordPress** alojado en un contenedor **Docker** en dos máquinas virtuales a través de un balanceador de carga y que, a su vez, estarán conectadas a recurso **Azure Database for SQL Server 2019**. Dicha conexión a SQL Server se realizará mediante **un service endpoint asociado a la VNet del entorno**, asegurando un acceso privado al recurso.

Cabe destacar, que el recurso Azure Database for SQL Server 2019 ya que, al tratarse de un servicio **serverless**, no es necesario realizar ninguna configuración adicional para que el servicio esté preparado para **alta disponibilidad y escalabilidad**.

A continuación, se muestra un esquema de alto nivel de la arquitectura planteada:



Para automatizar el despliegue de los servicios que habilitaremos en las máquinas virtuales, vamos a realizar un pequeño desarrollo en **Ansible**.

Además, para **optimizar el rendimiento** del servicio, vamos a incluir un balanceador de carga, brindando equilibrio a la carga del tráfico de la red privada y de internet con alto rendimiento y baja latencia. Lo realizaremos con **Azure Load Balancer**.

Y, por último, para poder tener una **recuperación ante desastres**, Azure nos brinda la posibilidad de tener los mismos servicios en una misma región, pero en distintas zonas de disponibilidad, para que, en caso de un desastre que afecte a una zona concreta, ya sea **de carácter físico o natural**, podremos seguir disponiendo de nuestro servicio en una zona alternativa.

1. Tecnologías empleadas.

Para llevar a cabo el proyecto vamos a hacer uso de varias tecnologías:

- **Servicio de nube pública:** Haremos uso de Microsoft Azure como nube pública, y dentro de la misma emplearemos los siguientes servicios:
 - o **Azure VMs:** emplearemos 2 máquinas virtuales alojadas cada una en una zona de la región diferente para la recuperación ante desastres.
 - o **Azure Database for SQL Server 2019:** para que el servicio de WordPress esté funcionando, necesitaremos de una base de datos, donde se alojarán los datos que nuestro servicio hará uso para guardar entradas, índices, imágenes, ...
 - o **Azure Load Balancer:** Azure nos pone a nuestra disposición un balanceador de carga que mejora la disponibilidad y la escalabilidad de la aplicación mediante la distribución del tráfico de red.
 - o **Azure Virtual Network:** Con este servicio, crearemos la red privada que conectará la instancia SQL Server con la máquina virtual en la que estará hospedado WordPress.
- **Docker:** el servicio de WordPress se servirá a través de un contenedor, cuya imagen la disponemos en [Docker Hub](#) de forma oficial.
- **Ansible:** la automatización del despliegue de los servicios será realizada a través de un pequeño desarrollo en ansible.
- **Terraform:** la automatización completa del escenario será realizada a través de esta herramienta, donde definiremos los recursos de la infraestructura.

2. Resultados esperados

Azure nos permite crear e implementar una gran variedad de aplicaciones, ya sean multimedia, de negocios, soluciones web,... Y debido a sus características, como el escalado automático, ya sea para aumentar o reducir en función de sus necesidades, al finalizar el despliegue del entorno, vamos a obtener una aplicación que tenga recuperación ante desastres alojada en una nube pública, con una base de datos segura y solo accesible desde su red interna y con un balanceador de carga para evitar una saturación debido a las peticiones que reciba de los diferentes clientes. Por ello, al finalizar, accederemos a nuestro dominio, y podremos acceder a nuestro WordPress y ver algunos posts que se albergan en la base de datos asignada.

2. ¿Qué es la nube?

2.1. ¿Qué es la informática en la nube?

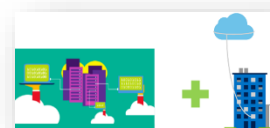
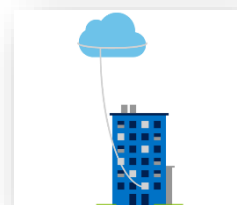
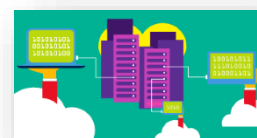
La informática en la nube es la prestación de servicios informáticos a través de Internet, lo que permite una innovación más rápida, recursos flexibles y precios regulables.



2.2. Tipos de nube

Existen tres tipos de nubes:

- **Nube pública:** necesita de un proveedor de servicio cloud o un proveedor de hosting, que proporciones recursos y servicios a múltiples organizaciones y usuarios. Se accede a través de una red segura.
- **Nube privada:** se trata de un entorno en la nube, pero en su centro propio de datos, donde es la empresa u organización la responsable de operar los servicios que brindan. La otra diferencia con la nube pública es que no proporciona acceso a los usuarios ajenos a la empresa u organización.
- **Nube híbrida:** combina nubes públicas y privadas que nos permitirán que las aplicaciones se ejecuten en la ubicación más adecuada para la misma.



2.3. Beneficios y consideraciones de la nube

La nube nos proporciona una serie de beneficios, tanto para individuos y empresas, entre los que se encuentran los siguientes:

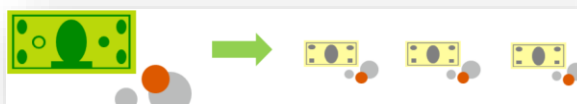
- **Alta disponibilidad:** debido a las regiones y las zonas en las que están alojados los centros de datos, el servicio tiene la capacidad de estar en funcionamiento y disponible de forma continua, incluso en situaciones de fallo y/o interrupciones, brindando también, **tolerancia a fallos**.

- **Escalabilidad y elasticidad:** nos permite ajustar en base a las necesidades, tanto de procesamiento como de almacenamiento, por lo que se puede escalar de forma horizontal y vertical sin necesidad de invertir en hardware.
- **Alcance global:** la distribución a nivel terráqueo de los centros de datos de los proveedores le concede a las empresas y organizaciones una presencia global sin la necesidad de invertir en infraestructura On-Premise.
- **Agilidad:** puede adaptarse de forma casi inmediata, según las necesidades de la empresa y/u organización, que va estrechamente ligada a la escalabilidad, al alcance, a la alta disponibilidad...
- **Capacidades de latencia del cliente:** esta capacidad es gracias a la replicación de datos en múltiples ubicaciones geográficas, a la implementación de redes de entrega de contenido (CDN) y a la implementación de servidores cercanos a los clientes físicamente (Edge servers).
- **Consideraciones de coste predictivo:** los servicios de nube pública cuentan con una calculadora de costo de recursos, por lo que antes de realizar una infraestructura podremos saber las cantidades necesarias para hacerla posible. ([Azure Pricing Calculator](#), [Google Cloud Pricing Calculator](#), [AWS Pricing Calculator](#)).

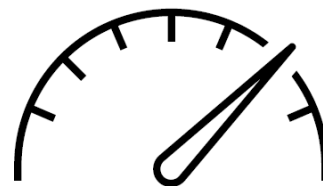
Alta disponibilidad	Tolerancia a errores
Escalabilidad	Elasticidad
Alcance global	Capacidades de latencia del cliente
Agilidad	Consideraciones de coste predictivo

En cuanto a las consideraciones que debemos de tener en cuenta de cara a realizar una migración a la nube, es importante tener en cuenta lo siguiente:

- Debemos tener en cuenta los gastos que nos podría causar una nube pública, privada o híbrida. Por ello, comparar entre **CapEx** (gastos de capital, es decir, el gasto inicial de una infraestructura física y los costes reducidos de cara a futuro) y **OpEx** (gastos operativos, es decir, los gastos derivados de los productos o servicios necesarios deducidos en el mismo año).

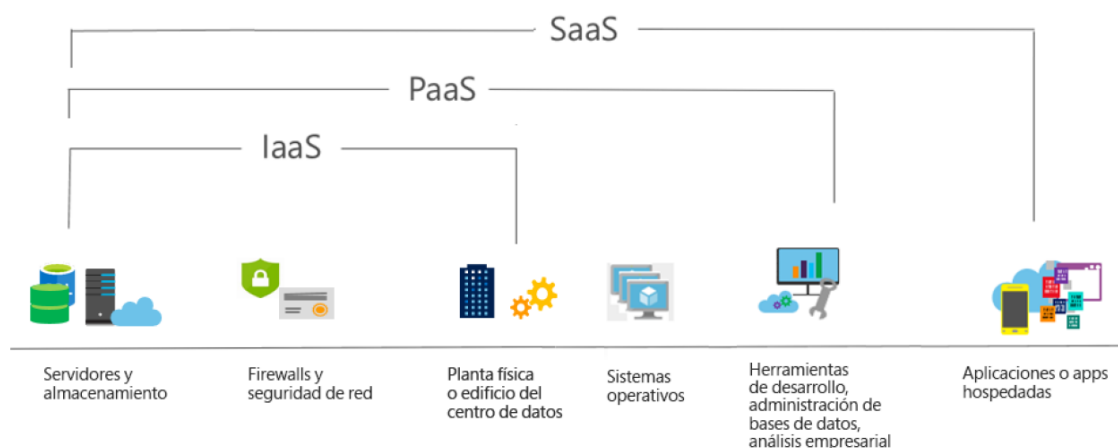


- También debemos considerar que es un modelo basado en el consumo, lo que significa que solo se paga por los recursos que se utilizan: **Lo que se usa es lo que se paga**. Como hemos comentado en las *consideraciones de coste predictivo* de este mismo apartado, a través del servicio de **Pricing Calculator** de las nubes públicas, podemos saber la facturación de uso real, tanto de forma total como por cada servicio y recursos.



2.4. Servicios en la nube

Como hemos visto a lo largo del curso, los servicios de la nube le permiten a la empresa y/u organización acceder a recursos de forma escalable y flexible, sin una inversión costosa de infraestructura física.



Además de estos servicios que podemos ver en la imagen anterior, hay servicios en la nube como lo son la base de datos como servicio (**DBaaS**) y la seguridad como servicio (**SECaaS**) que proporcionan a las empresas estos recursos de forma más específica y especializada.

En conclusión, los servicios de la nube son flexibles y escalables para que así, las empresas y/u organizaciones puedan acceder a recursos informáticos y que, además de los servicios que estamos acostumbrados a encontrarnos en la nube, también podemos encontrarnos con servicios especializados como los que hemos nombrado anteriormente.

3. ¿Qué es Terraform?

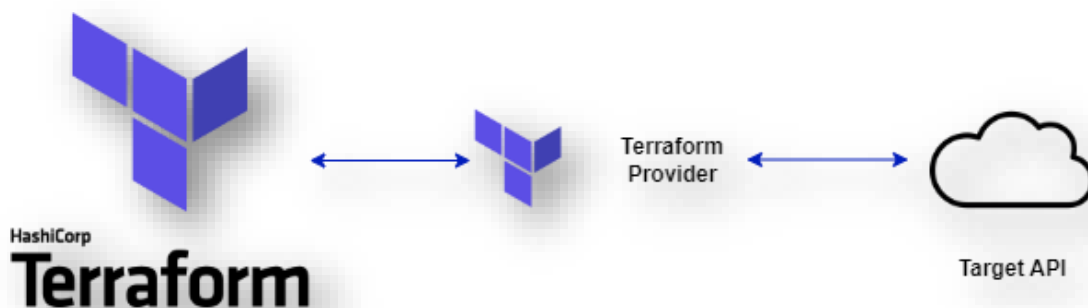
Terraform es una herramienta de infraestructura como código (IaC) que permite a los equipos de desarrollo y operaciones definir, crear y administrar la infraestructura de forma programática. Con Terraform, los desarrolladores pueden definir la infraestructura en un archivo de configuración y luego desplegarla de manera automatizada y reproducible.

El proceso de trabajo con Terraform comienza con la definición de la infraestructura en un archivo de configuración escrito en el lenguaje de configuración (YAML) de Terraform. Luego, lo analiza y determina los recursos que necesitan ser creados o actualizados en función de la definición proporcionada. Por último, aplica los cambios en la infraestructura en función de la definición proporcionada.

Terraform es compatible con varios proveedores de nube, incluyendo Amazon Web Services, Microsoft Azure, Google Cloud Platform, y muchos otros. También es posible usar Terraform para administrar infraestructura local, como servidores físicos o virtuales.

3.1. ¿Cómo funciona Terraform?

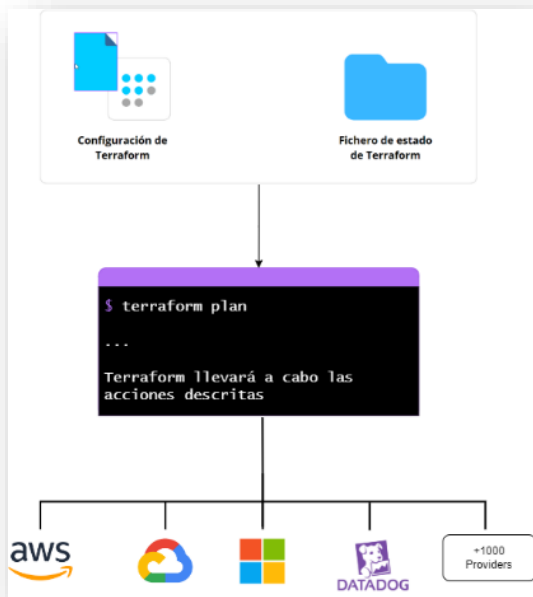
Terraform crea y administra recursos en plataformas cloud y otros servicios a través de sus APIs (Application Programming Interface). Los providers habilitan a Terraform para trabajar de forma virtual con cualquier plataforma o servicio accesible a través de una API.



Hashicorp y la comunidad de Terraform tienen actualmente miles de providers para administrar diferentes tipos de recursos y servicios, y todos ellos los podemos encontrar de forma pública en [Terraform Registry](#), donde podremos encontrar Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub y muchos más.

Realmente, el workflow (ciclo de trabajo) de Terraform consiste en cinco pasos:

- **Escribir:** Definimos los recursos, el o los servicios de nube pública empleados y los providers necesarios.



- **Formatear:** reescribimos los archivos de configuración de Terraform a un formato y estilo normalizado.
- **Validar:** Antes de ejecutar nuestra configuración, validaremos la sintaxis y se realiza en el directorio que contiene los archivos de configuración de Terraform.
- **Plan:** Terraform crea un plan de ejecución en el que se describe la infraestructura que creará, destruirá o actualizará en función de la ya existente y su configuración (en caso de ya existir una infraestructura desplegada).
- **Apply:** Una vez realizadas todas las comprobaciones, Terraform llevará a cabo las configuraciones ya escritas, siempre respetando la dependencia de recursos.

3.2. Conceptos claves de Terraform

A continuación, voy a presentar algunos de los conceptos claves referentes a Terraform. Nos serán útiles y de ayuda a la hora de comprender un poco mejor esta herramienta de automatización de infraestructura.

- **Infraestructura como código (IaC):** Es una metodología de gestión y provisionamiento de una infraestructura mediante código, en lugar de realizar la configuración de forma manual. Es decir, que a través de scripts y ficheros de configuración para automatizar la creación y la gestión de infraestructuras. Esto hace que se facilite la creación, el mantenimiento y la reproducción de infraestructuras en diferentes entornos.
- **Configuración:** Está formada por un conjunto de ficheros de configuración, en los que se describen los recursos que se van a crear y para ello podemos realizarlo en formato **JSON** o en **HCL** (Hashicorp Configuration Language).
- **Proveedor (providers):** Es una API implementada dentro de Terraform que nos permite relacionarse con los distintos servicios en la nube o de infraestructuras. Algunos de estos proveedores son AWS, Google Cloud Platform, Azure, etc.... Los podemos encontrar en [Browse Providers](#) en su página oficial.

- **Recurso:** Es una instancia individual, ya sea de un servicio o de un propio recurso que se vaya a crear, actualizar o eliminar, como lo sería una máquina virtual en AWS o una instancia EC2 en AWS.
- **Módulo:** es una colección reutilizable de configuraciones y recursos definidos por separado, permitiendo organizar la infraestructura en bloques lógicos. Se describen en ficheros con extensión **.tf**.
- **Estado:** El estado de Terraform es un archivo que describe la infraestructura actual y permite a Terraform hacer un seguimiento de los cambios realizados en la infraestructura a lo largo del tiempo.
- **Plan:** Es la forma de Terraform de representar los cambios realizados o que serán realizados al ejecutar Terraform, por lo que nos brinda la oportunidad de revisar, antes de que se apliquen, los cambios.

3.3. Cómo se instala en WSL (Windows Subsystem for Linux)

Para instalar WSL, lo haremos desde la [Microsoft Store](#). Una vez instalado, procederemos a la instalación de **Terraform**:

1. Nos aseguramos de tener los paquetes necesarios para la instalación de Terraform:

```
sudo apt-get update  
sudo apt-get upgrade
```

2. Nos descargamos el fichero de la clave pública GPG de HashiCorp, la desencriptamos y la pasamos de formato para que pueda ser empleada por el sistema y la guardamos en la ruta que aparece en el comando, para que posteriormente, se pueda verificar la autenticidad de los paquetes para que los paquetes de Terraform puedan ser descargados desde los repositorios de Hashicorp.

```
wget -O- https://apt.releases.hashicorp.com/gpg | gpg --  
dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-  
keyring.gpg
```

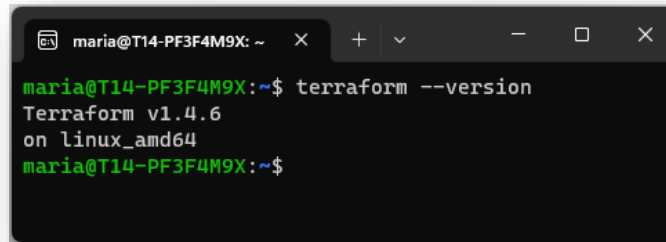
3. Añadimos los repositorios que tomarán la firma de verificación del directorio que asignamos anteriormente. También deberemos añadir la ruta del repositorio de Hashicorp para que se puedan descargar e instalar los paquetes desde el repositorio.

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-  
keyring.gpg] https://apt.releases.hashicorp.com  
$(lsb_release -cs) main" | sudo tee  
/etc/apt/sources.list.d/hashicorp.list
```

4. Por último, actualizamos para que se efectúen los cambios en el fichero donde están alojados los repositorios y procedemos a la instalación de Terraform

```
sudo apt update && sudo apt install terraform
```

Y como podremos comprobar, al ejecutar `terraform --version` vemos que está instalado correctamente.



```
maria@T14-PF3F4M9X: ~  
maria@T14-PF3F4M9X:~$ terraform --version  
Terraform v1.4.6  
on linux_amd64  
maria@T14-PF3F4M9X:~$
```

4. ¿Qué es Azure?

Azure es la plataforma en la nube de Microsoft que ofrece servicios de infraestructura, plataforma y software como servicio (IaaS, PaaS y SaaS) para alojar, gestionar y desarrollar aplicaciones y servicios en la nube. La plataforma es flexible y escalable, permitiendo a los usuarios escalar recursos automáticamente para satisfacer las demandas cambiantes de sus aplicaciones y servicios. Además, Azure se integra con una amplia gama de herramientas y servicios de terceros y ofrece características de seguridad como autenticación y autorización, cifrado de datos y cumplimiento normativo para proteger los datos y las aplicaciones alojados en la plataforma.

En resumen, Azure es una solución completa y confiable para las necesidades de infraestructura y aplicaciones en la nube de las empresas.

4.1. ¿Cómo funciona Azure?

Microsoft Azure, como cualquier plataforma en la nube, se basa en lo que conocemos como virtualización, y la emplea para ejecutar hardware virtualizado como si se tratase de hardware real.

La nube está formada por un conjunto de servidores que podremos encontrarnos en uno o varios centros de datos, donde tienen una colección de servidores colocados en bastidores que, a su vez, están agrupados en clústers que se emplearán para ejecutar los recursos que el usuario haya configurado para su infraestructura.

Por ello, Microsoft tiene distribuido a lo largo de todo el mundo, regiones y zonas habilitadas para su servicio. Como podemos ver en la siguiente imagen, existen regiones de próxima apertura, como lo son los centros de Madrid, Algete y Meco, sin fecha aún, pero ya acordado entre Ferrovial y Microsoft.



Zonas y regiones de Azure


Azure ofrece más regiones a nivel global que cualquier otro proveedor en la nube, ya que cuenta con más de 60 regiones representadas en más de 140 países. Las regiones se componen de uno o más centros de datos próximos.


Entre los pares de regiones, debe existir una separación de 300 millas (o lo que es lo mismo, casi 500 kms). Las actualizaciones se implementan de forma secuencial, para que de esta forma se minimicen los tiempos de inactividad. En caso de una interrupción en la región, se priorizará su recuperación. En el sitio oficial de Microsoft podemos verlo en [Replicación entre regiones en Azure: continuidad empresarial y recuperación ante desastres](#).


4.2. Conceptos claves de Azure


A continuación, voy a presentar algunos de los conceptos claves referentes a Microsoft Azure. Nos serán útiles y de ayuda a la hora de comprender un poco mejor este servicio de la nube.


- **Recurso:** Es cualquier elemento que podamos implementar y administrar, como, por ejemplo, máquinas virtuales, bases de datos, aplicaciones web, redes virtuales...



 Máquinas virtuales

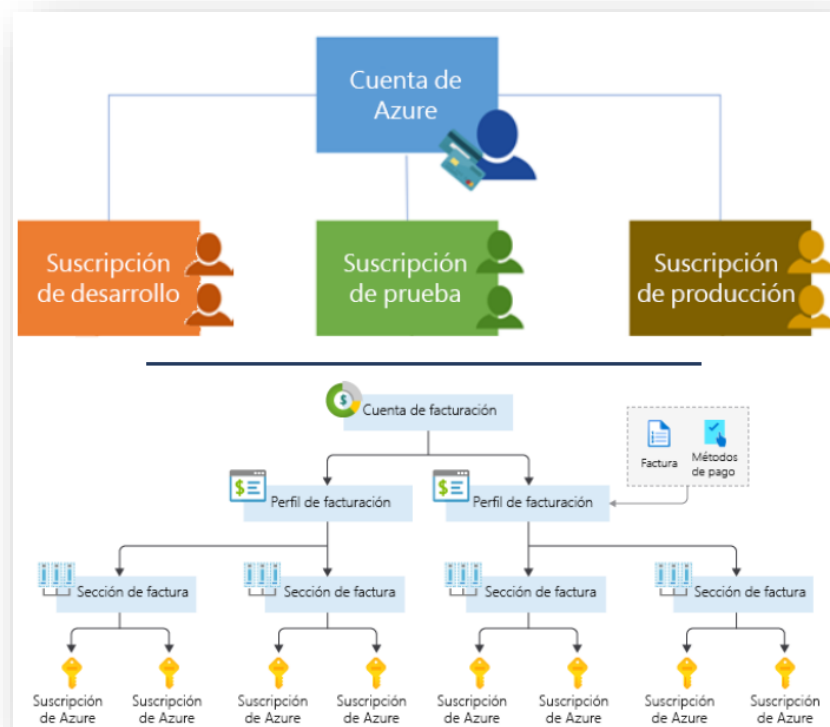

 Cuentas de almacenamiento


 Redes virtuales
- **Suscripción:** Los recursos de Azure están asociados a una suscripción, por lo que, tener una, es el primer paso para empezar en Azure. Posee límite de facturación, que genera informes de facturación y facturas independientes para cada suscripción.


 App Services


 SQL Database


 Funciones



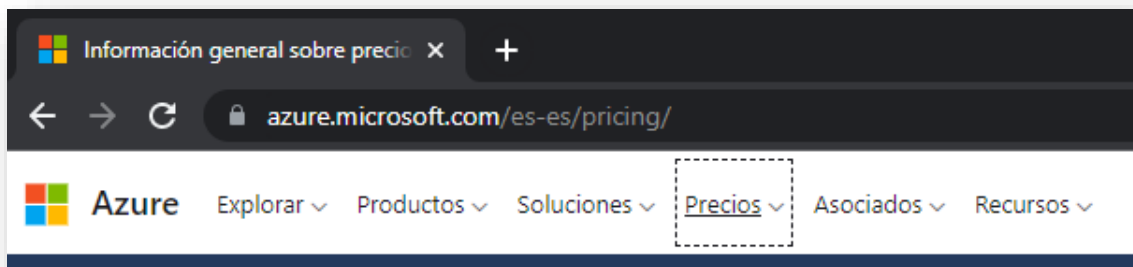
- **Cuenta de Azure:** Una cuenta de Azure está ligada a un correo electrónico, que será la que se convierte en la cuenta principal de Azure, siendo responsable de los pagos mensuales de los recursos empleados por la suscripción. Por ello, cuando la creamos, nos pedirá que proporcionemos desde información de contacto, facturación hasta documentación de identificación.
- **Azure Active Directory (AD):** Se trata del servicio que administra el acceso y las identidades de Microsoft, para permitir el inicio de sesión y acceso a los recursos.
- **Grupo de recursos (Resources Group):** Es un contenedor lógico que organiza y administra los recursos relacionados de Azure. Además, pueden aplicárseles etiqueta, políticas de seguridad, costo de los recursos... Los grupos de recursos se pueden ubicar en distintas regiones de Azure, permitiendo flexibilidad.
- **Región:** Es la zona geográfica donde se encuentran los centros de datos de Azure, siendo cada una independiente y con su propia red de centros de datos aislada.



4.3. Precios de Azure

Azure es una plataforma con una variedad amplia de servicios, a lo igual que sus planes de precios, que se pueden adaptar a las diferentes necesidades y presupuestos. Los servicios empleados de forma más común y sus precios más aproximados son los siguientes:

- **Máquinas virtuales:** Dependiendo de la configuración y de su capacidad, su precio oscila entre los 12€ al mes, pudiendo llegar a varios miles de dólares al mes.
- **Almacenamiento:** Éste dependerá del tipo de almacenamiento y de la cantidad de datos que se almacenen. El precio puede variar entre los 0.008 € por GB hasta varios miles de euros por GB al mes.
- **Bases de datos:** Dependiendo del tamaño y del tipo de la base de datos, el precio mínimo sería aproximadamente de 3€ al mes hasta varios cientos de euros al mes.
- **Redes:** Los precios variarán según la zona geográfica y el uso de estas, aunque el coste puede oscilar entre los 0.008€ y los 0.08€ por GB de tráfico.



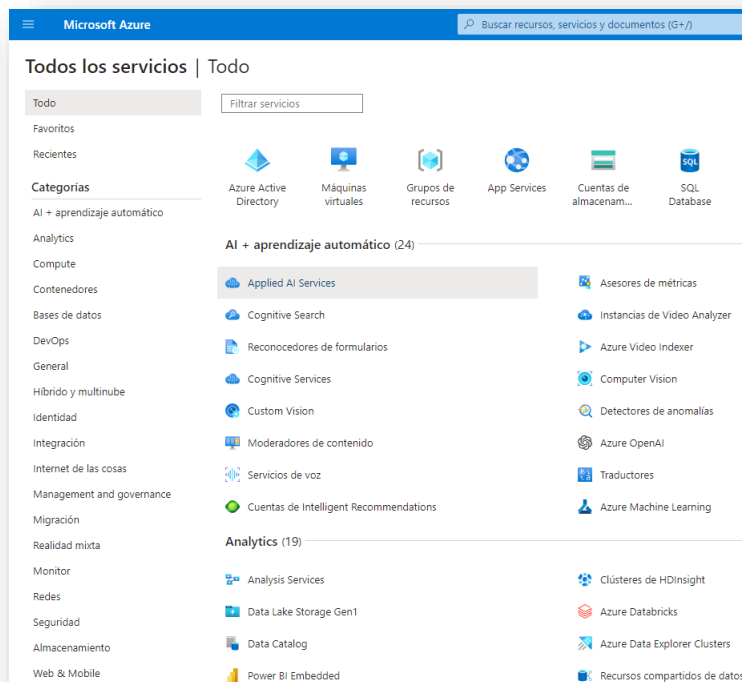
Para más información acerca de los precios, podemos encontrar en la propia página oficial de Azure, si hacemos clic en el botón de “Precios” y accederemos a una lista detallada de los servicios de los que disponemos en Azure y sus correspondientes precios. Para mayor facilidad, podemos entrar en [Precios de Cloud Services](#).

4.4. ¿Cómo podemos usar Azure?

Podemos hacer uso de Azure de dos formas distintas:

4.4.1. Azure Portal

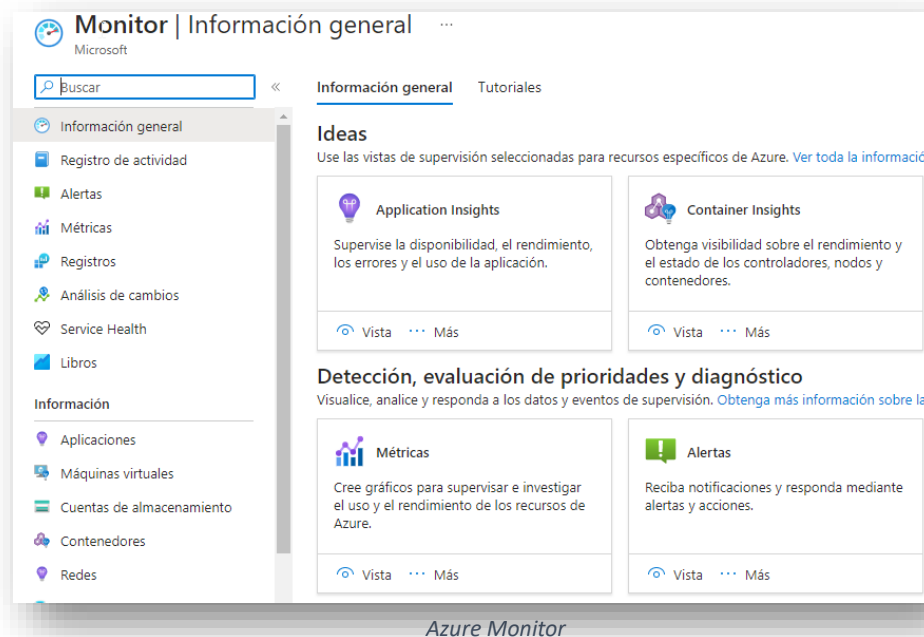
La primera es a través del portal propio de [Microsoft Azure](#). Es una plataforma en línea donde los usuarios de Azure pueden administrar y monitorear los recursos y servicios. A través de su interfaz gráfica, se nos facilita la gestión de aplicaciones y servicios en la nube de Azure.



Microsoft Azure Portal

Desde aquí, nosotros como usuarios podemos crear, configurar, monitorear todo tipo de recursos que nos ofrece Azure: máquinas virtuales, redes, aplicaciones web, bases de datos, balanceadores de carga, ... Otra de las características que tiene es que podemos

configurar la seguridad de nuestros recursos, ver el uso y rendimiento de nuestros servicios, a través de Azure monitor, los costes de los servicios, etc.



Esta herramienta es muy versátil y poderosa, tanto para desarrolladores como para administradores, ya que pueden administrar de forma fácil y eficiente los recursos en la nube. Cabe decir que Azure también cuenta con la integración de Azure DevOps y Active Directory, facilitando la integración y la gestión de recursos en la nube.

4.4.2. Azure CLI

Azure Command-Line Interface (Azure CLI) se trata de una herramienta de la línea de comandos que nos permitirá a los usuarios administrar e interactuar con los servicios que nos ofrece Azure, pero de forma programática.

Admite variedad de sistemas operativos y es compatibles con numerosos lenguajes como Bash, JavaScript, Python... Otras de las ventajas de hacer uso de Azure CLI es poder automatizar las tareas de administración de Azure.

También se integran las herramientas de Azure DevOps y Azure PowerShell, para de esa manera permitir a los usuarios crear workflows (flujos de trabajo automatizado) y/o administrar los recursos de forma eficiente.

Para instalar Azure CLI, lo podemos hacer tanto en entornos Windows, MacOS y Linux. Como podemos leer en la documentación de la [instalación de la CLI de Azure](#), también podríamos hacerlo en un contenedor Docker y Azure Cloud Shell.

4.4.2.1. Instalación de Azure CLI en Windows

La instalación podemos hacerla de 3 formas: a través del [instalador de Microsoft \(MSI\)](#) tanto la última versión como una versión específica, [por el instalador de Microsoft \(MSI\) con comando](#) o a través del [administrador de paquetes de Windows](#) (aunque esta última nos instalará la versión preliminar).

A la hora de ejecutar la CLI, lo podremos hacer desde PowerShell o a través del símbolo de sistema de Windows.

En caso de que queramos realizar la desinstalación, en el apartado de [desinstalación](#) nos vienen los pasos completos para su realización.

4.1.1.1. Instalación de Azure CLI en MacOs

La instalación en un Sistema MacOs se realizará a través de Hombrew como nos indican en el apartado de [instalación de la CLI de Azure en macOS](#), donde nos podemos encontrar soluciones de problemas frecuentes referente a la instalación de Azure CLI.

En caso de que queramos realizar la desinstalación, en el apartado de [desinstalación](#) nos vienen los pasos completos para su realización.

4.4.3. A través de herramientas de IaaS (Infrastructure as a Code)

Otra forma en la que también podemos hacer uso de Azure es a través de herramientas de IaaS como lo serían Ansible, Terraform, Puppet, Chef... Todas ellas permiten poder definir y administrar los recursos de una infraestructura a través de ficheros de configuración y scripts.

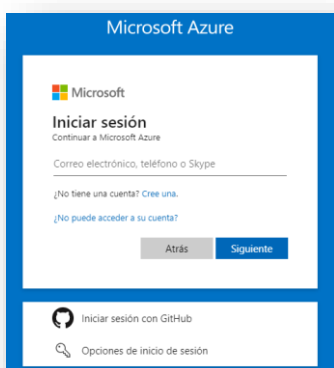
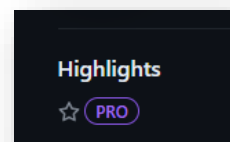
Al hacer uso de IaaS, el tiempo de la implementación se reduce, los errores de configuración se minimizan y la eficiencia en cuanto a la administración de recursos en la nube aumenta, sin olvidar que también nos permiten el control y la gestión de versiones, facilitando las colaboraciones y el trabajo en equipo en proyectos.

En mi caso, he empleado Terraform, cuya instalación la podemos encontrar en el epígrafe titulado [Cómo se instala en WSL \(Windows Subsystem for Linux\)](#).

5. Obtención de cuenta en Azure

Como hemos podido leer en el apartado [Precios de Azure](#), un servicio de nube no es barato, por lo que, para realizar este proyecto he podido acceder a una suscripción de un año de Azure y vamos a ver cómo podemos obtenerla.

Pero, primero, lo que debemos conseguir es una cuenta de [GitHub Education](#). Al no poseer correo electrónico educativo, GitHub nos permite subir una foto o un justificante de escolarización o matrícula en un centro de estudios. En el plazo de 2 horas, deberíamos tener nuestra cuenta verificada.



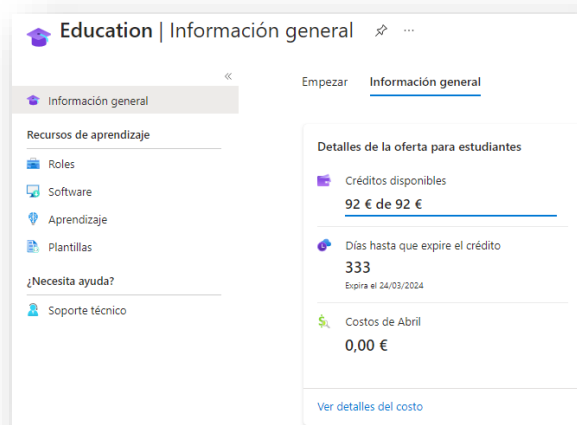
La cuenta de GitHub Education tiene como beneficio una suscripción de Azure. Tras la compra de Microsoft de GitHub, podemos ingresar en [Microsoft Azure](#) con nuestra cuenta de GitHub.

Y una vez elegida la opción “Iniciar sesión con Github” podremos tener acceso a nuestra cuenta de Microsoft Azure for Students. Algunos de sus beneficios son:

- **Azure Virtual Machines: Linux:** 750 horas de máquinas virtuales ampliables B1s.
- **Azure SQL Database:** Instancia S0 de 250 GB con 10 unidades de transacción de base de datos.
- **Azure Blob Storage:** 5 GB de almacenamiento con LRS (redundancia local) en bloques de acceso frecuente, con 20000 operaciones de lectura y 10000 de escritura.
- **Azure Functions:** 1 millón de solicitudes.
- Y más...

Muchos de los beneficios son para siempre y otros sólo tienen una duración de 12 meses. Sin el algún momento necesitamos de un servicio de pago, tendríamos 92€de crédito.

Para más información acerca de todos los beneficios que nos brinda, solo tenemos que dirigirnos a la página de [Azure for Students](#).



6. Parte práctica

El escenario desde el que se va a desplegar el proyecto va a ser en un entorno local, desde donde lanzaremos los scripts y los comandos a través de la consola de WSL, realizando así el despliegue de una infraestructura en Microsoft Azure.

Para mayor comprensión del entorno y del proyecto en sí, en el siguiente [repositorio](#) se encontrará la guía y el código completo tanto de Ansible como de Terraform. De forma más gráfica, podemos verlo en el epígrafe de este mismo documento llamado [Descripción](#).

6.1. Objetivos

El principal objetivo de la ejecución de este escenario es tener un servicio desplegado de forma automática, con un balanceador de carga, recuperación ante desastres y alta disponibilidad. Este servicio será accedido a través del dominio mariatec.es mediante la configuración en el servidor DNS y así, poder visualizar un servicio WordPress totalmente alojado en la nube de Microsoft Azure.

6.2. Requisitos

Para poder realizar el despliegue completo necesitaremos:

- Suscripción de Microsoft Azure. (Ya explicado en el apartado [Obtención de cuenta en Azure](#) en caso de cuenta de estudiante).
- Service Principal configurado en la suscripción de Azure.
- Windows Subsystem for Linux (WSL) con las siguientes herramientas instaladas:
 - o Azure CLI.
 - o Ansible.
 - o SSHPass.
 - o Terraform.

6.3. Pasos a seguir.

Una vez preparado nuestro escenario, nos disponemos a realizar nuestro despliegue continuo. Para ello, vamos a seguir los siguientes pasos:

1. Desde Azure Portal, se accede a Azure Active Directory (AD) y nos dirigimos en el menú lateral izquierdo a **Registro de aplicaciones**. A continuación, creamos un **+ Nuevo registro**. En la siguiente pantalla, indicamos nombre y **Tipo de cuenta compatible**, seleccionamos **Solo cuentas de este directorio organizativo**. Para finalizar, pulsamos en Registrar.
2. En la siguiente pantalla, en el menú del lateral izquierdo, accedemos a **Certificados y secretos**. Una vez aquí, en **Secretos del cliente** añadiremos un **+ Nuevo secreto de cliente** que será empleado por Terraform para autenticarse en Azure.

3. Añadimos una descripción y el tiempo de expiración y agregamos.

Agregar un secreto de cliente ✕

Descripción

Expira

Agregar
Cancelar

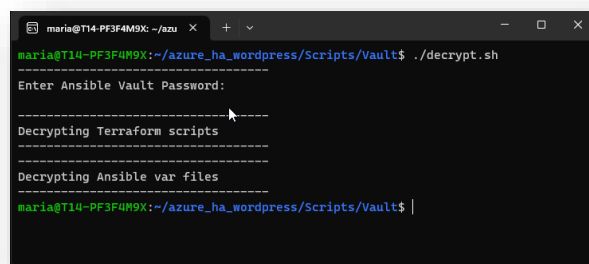
4. Es importante guardar de forma meticulosa los valores de **Valor** e **Id del secreto**, ya que estos serán utilizados más adelante.
5. En información general, será necesario guardas de la misma forma, los campos **Id de aplicación** (cliente), **Identificador de objeto** e **Id del directorio**, ya que serán utilizados posteriormente.

Una vez finalizado esto, comenzamos con la parte de **Terraform**.

6. Clonamos el repositorio donde tenemos alojado el código de nuestra infraestructura: el código de Terraform, de Ansible y los scripts necesarios para ello.

```
git clone https://github.com/Legnakra/azure_ha_wordpress.git
```

7. Una vez clonado, dentro de la carpeta del repositorio, nos dirigimos a Scripts/Vault, donde se encontrarán los dos scripts encargados de encriptar y desencriptar los archivos con datos sensibles, los cuales se han protegido haciendo uso de Ansible Vault, el cual emplea AES256 como algoritmo.
8. Tras esto, ejecutamos primero el script *decrypt.sh*, el cual solicitará una contraseña y dejará disponible los ficheros para su uso.



9. A continuación, acudimos al directorio *Scripts/Terraform*, donde están alojados los tres scripts encargados de operar la plataforma. Los scripts son los siguientes:
 - *plan.sh*: Será el encargado de inicializar el directorio del entorno de Terraform y mostrar la salida del comando `terraform plan`, el cual mostrará los cambios que realizará en el entorno de Azure.
 - *apply.sh*: Este nos permitirá desplegar los cambios mostrados por el script anterior.
 - *destroy.sh*: Por último, este permitirá destruir el entorno al completo.

10. Ejecutamos el script *plan.sh*.

```

maria@T14-PF3F4M9X:~/azul/azul_scripts/Terraform$ ./plan.sh
-----
Configuring Azure subscription
-----

-----
Launching Terraform plan
-----

Initializing the backend...

Successfully configured the backend "local"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing modules...
- database in ../modules/database
- load-balancer in ../modules/load-balancer
- networking in ../modules/networking
- resource_group in ../modules/resource-group
- storage in ../modules/storage
- virtual-machines in ../modules/virtual-machines

Initializing provider plugins...
- Finding hashicorp/azurerm versions matching "3.53.0"...
- Installing hashicorp/azurerm v3.53.0...
- Installed hashicorp/azurerm v3.53.0 (signed by HashiCorp)

Terraform has created a lock file ".terraform.lock.hcl" to record the provider
selections it made above. Include this file in your version control repository
    
```

```

maria@T14-PF3F4M9X:~/azul/azul_scripts/Terraform$ ./plan.sh
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# module.database.azuremysql_database.wp-mysql will be created
+ resource "azurerm_mysql_database" "wp-mysql" {
+   charset      = "utf8"
+   collation    = "utf8_general_ci"
+   id           = (known after apply)
+   name         = "wordpress"
+   resource_group_name = "rg-hawordpress-westeu-001"
+   server_name  = "mysql-hawordpress-westeu-001"
}

# module.database.azuremysql_firewall_rule.mysql_azure_vms will be created
+ resource "azurerm_mysql_firewall_rule" "mysql_azure_vms" {
+   end_ip_address = "0.0.0.0"
+   id             = (known after apply)
+   name           = "azure-access"
+   resource_group_name = "rg-hawordpress-westeu-001"
+   server_name    = "mysql-hawordpress-westeu-001"
+   start_ip_address = "0.0.0.0"
}

# module.database.azuremysql_server.wp-mysql will be created
+ resource "azurerm_mysql_server" "wp-mysql" {
    
```

11. Ejecutamos *apply.sh*.

```

maria@T14-PF3F4M9X:~/azul/azul_scripts/Terraform$ ./apply.sh
-----
Configuring Azure subscription
-----

-----
Deploying Terraform resources
-----

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# module.database.azuremysql_database.wp-mysql will be created
+ resource "azurerm_mysql_database" "wp-mysql" {
+   charset      = "utf8"
+   collation    = "utf8_general_ci"
+   id           = (known after apply)
+   name         = "wordpress"
+   resource_group_name = "rg-hawordpress-westeu-001"
+   server_name  = "mysql-hawordpress-westeu-001"
}

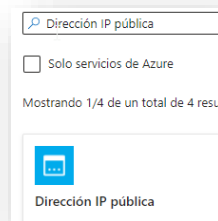
# module.database.azuremysql_firewall_rule.mysql_azure_vms will be created
+ resource "azurerm_mysql_firewall_rule" "mysql_azure_vms" {
    
```

```
1731e67aae9/resourceGroups/rg-hawordpress-westeu-001/providers/
databases/wordpress]
module.database.azurearm_mysql_firewall_rule.mysql_azure_vms: C
-4754-bc57-31731e67aae9/resourceGroups/rg-hawordpress-westeu-6
-westeu-001/firewallRules/azure-access]

Apply complete! Resources: 23 added, 0 changed, 0 destroyed.
```

Ya desplegada nuestra infraestructura, como podemos visualizar en Azure Portal, procederemos a la creación y configuración de las IPs públicas que irán asociadas a las tarjetas de red de cada una de las máquinas virtuales de nuestra infraestructura.

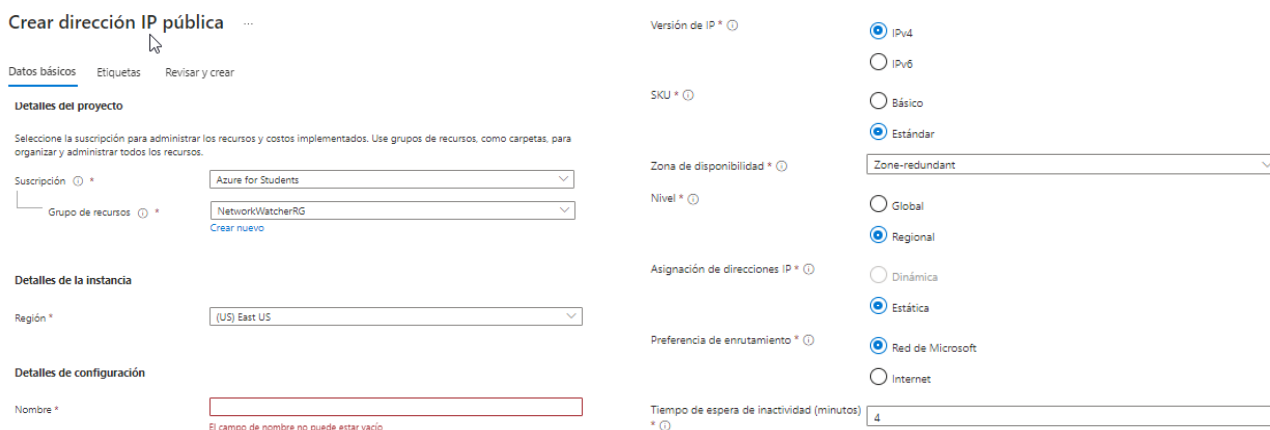
12. Dentro de nuestro grupo de recursos (resource group), nos aparece en el banner superior un botón **+ Crear**. Aquí, en la barra de búsqueda, escribimos *IP pública* y seleccionamos *Dirección IP pública*.



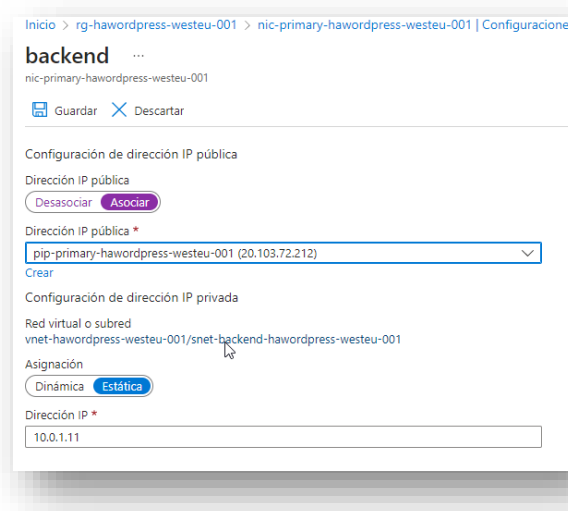
13. Se nos abrirá la siguiente pantalla, donde tendremos que configurar los siguientes parámetros:

- a. Tipo de suscripción.
- b. Grupo de recursos.
- c. Región en la que estará alojada la instancia.
- d. Nombre de la red.
- e. Versión IP.
- f. SKU.
- g. Disponibilidad.
- h. Nivel.
- i. Asignación de IP.
- j. Enrutamiento.
- k. Tiempo de espera de inactividad.

Estos parámetros vienen configurados de forma predeterminada, por lo que no será necesario realizar modificación alguna.



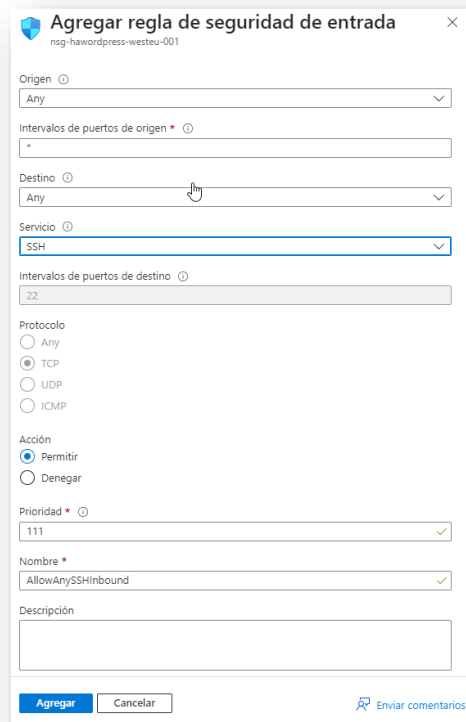
Los pasos 12 y 13 los realizaremos dos veces: una por cada máquina virtual, en las que habrá una interfaz de red en cada una. Como podemos ver en la siguiente imagen, una vez creadas las IPs, las asociamos a las tarjetas de red de cada una de las máquinas.



Esto se realiza ya que, a pesar de que el balanceador de carga dispone de una IP pública, para poder configurar cada una de las máquinas virtuales con Ansible, necesitaremos acceder de manera individual a cada una de ellas.

- 14. El entorno dispone de un grupo de seguridad de red asociado actuando como una suerte de firewall en el cual, Terraform despliega por defecto los puertos 80 y 442 como accesibles desde Internet.

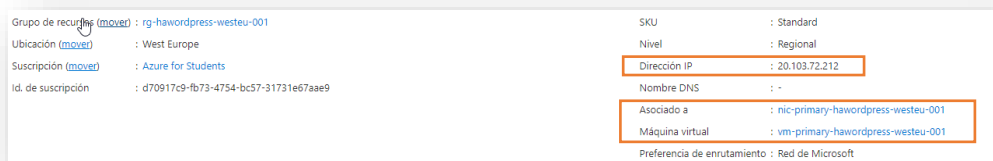
Dado que Ansible hace uso del protocolo SSH para realizar cambios, añadiremos de manera temporal el puerto 22 como accesible de forma pública.



A continuación, podemos visualizar las reglas activas en el grupo de seguridad. Si bien también podemos ver que la regla que hemos configurado para el puerto 22 nos aparece con una advertencia. Esto es debido a que Azure nos indica que dicha regla es potencialmente peligrosa. Por ahora ignoraremos esta advertencia ya que la regla se eliminará una vez sea completado el despliegue de Ansible.

<input type="checkbox"/>	100	http	80	Tcp
<input type="checkbox"/>	101	https	443	Tcp
<input type="checkbox"/>	111	AllowAnySSHInbound	22	TCP

15. Nos dirigiremos a las dos IPs públicas que hemos creado y conservamos ambas IP que copiaremos en el fichero hosts que se encontrará dentro del directorio Ansible/inventories.



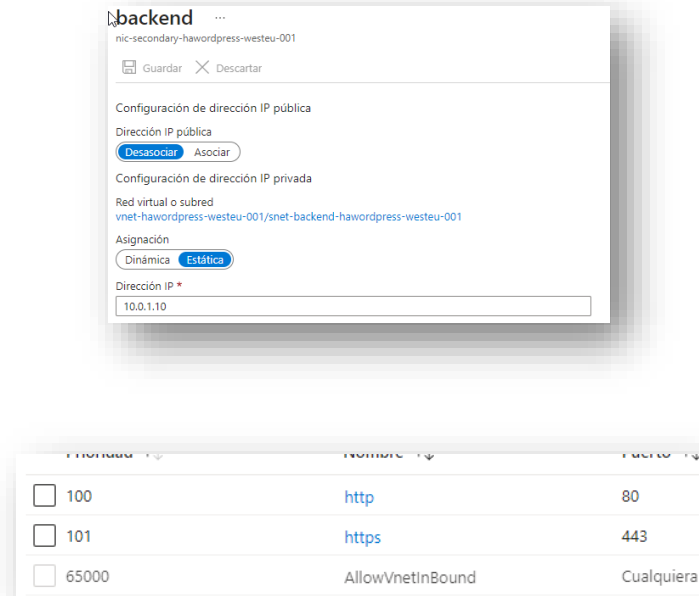
```
maria@T14-PF3F4M9X:~/a
[ha_wordpress]
20.103.72.212
20.103.73.157
```


16. Una vez realizado el cambio, realizaremos el despliegue de Ansible.

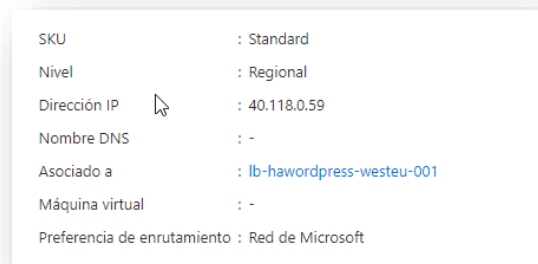
```
maria@T14-PF3F4M9X: ~/azu
maria@T14-PF3F4M9X:~/azure_ha_wordpress/Scripts/Ansible$ ./deploy.sh
Deploying Wordpress
-----
PLAY [ha_wordpress] *****
TASK [Gathering Facts] *****
ok: [20.103.73.157]
ok: [20.103.72.212]
TASK [docker-install : Actualizar la maquina] *****
changed: [20.103.73.157]
changed: [20.103.72.212]
TASK [docker-install : Instalar aptitude] *****
changed: [20.103.73.157]
changed: [20.103.72.212]
TASK [docker-install : Instalar prerequisites] *****
changed: [20.103.73.157]
changed: [20.103.72.212]
TASK [docker-install : Agregar clave GPG de los repositorios de Docker] *****
changed: [20.103.73.157]
changed: [20.103.72.212]
TASK [docker-install : Agregar repositorio oficial de Docker] *****
```

```
maria@T14-PF3F4M9X: ~/azu
TASK [wordpress : Esperar 5 segundos para inicializar Wordpress] *****
Pausing for 5 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [20.103.72.212]
TASK [wordpress : Ejecutar instalacion automatica de Wordpress] *****
skipping: [20.103.73.157]
changed: [20.103.72.212]
TASK [wordpress : Esperar a que finalice la instalacion] *****
Pausing for 60 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [20.103.72.212]
TASK [wordpress : Obtener estado de la instalacion] *****
skipping: [20.103.73.157]
ok: [20.103.72.212]
TASK [wordpress : Comprobar si la instalacion ha terminado correctamente] *****
skipping: [20.103.72.212]
skipping: [20.103.73.157]
TASK [wordpress : Eliminar contenedor de Wordpress CLI] *****
skipping: [20.103.73.157]
changed: [20.103.72.212]
TASK [wordpress : Eliminar imagen de Wordpress CLI] *****
skipping: [20.103.73.157]
changed: [20.103.72.212]
PLAY RECAP *****
20.103.72.212 : ok=9  changed=3  unreachable=0  failed=0  skipped=1  rescued=0  ignored=0
20.103.73.157 : ok=3  changed=0  unreachable=0  failed=0  skipped=5  rescued=0  ignored=0
maria@T14-PF3F4M9X:~/azure_ha_wordpress/Scripts/Ansible$ |
```

17. Una vez ejecutado correctamente el playbook, se desasociarán y eliminarán las direcciones IPs públicas temporales de las dos máquinas virtuales, y se eliminará la regla que permite el tráfico por el puerto 22 del grupo de seguridad de red.



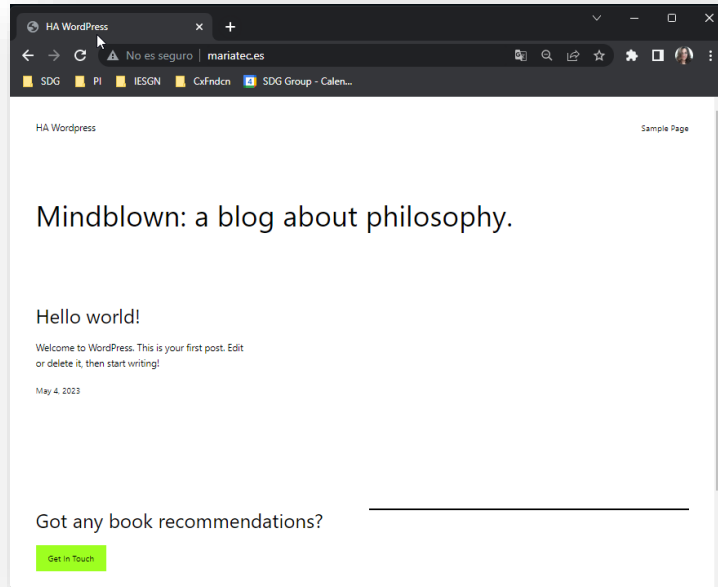
18. Una vez realizado exitosamente, procederemos a tomar la IP que se nos muestra en el balanceador de carga.



19. Esta dirección la necesitaremos para configurar un registro DNS para que, cuando accedamos a mariatec.es nos redirija a la página principal de nuestro WordPress desplegado en Azure con Terraform y Ansible.



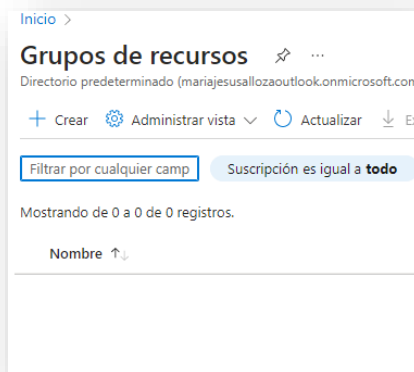
20. Una vez hemos esperado a que los servidores DNS actualicen los registros, si accedemos mediante cualquier navegador a la dirección mariatec.es podremos comprobar que disponemos del entorno WordPress perfectamente desplegado y configurado en alta disponibilidad.



NOTA: Este entorno se puede configurar mediante SSL haciendo uso de https, pero dado que el objetivo principal del proyecto radica en mostrar la flexibilidad de la nube y las capacidades del trabajo conjunto de Ansible y Terraform, se ha decidido no incluir dicha configuración en el proyecto.

21. Para finalizar, procederemos a ejecutar el script *destroy.sh* alojado en Scripts/Terraform cuya función será destruir por completo toda la infraestructura desplegada.

```
module.resource_group.azure_rm_resource_group.resource-group: Destroyin
31e67aae9/resourceGroups/rg-hawordpress-westeu-001]
module.resource_group.azure_rm_resource_group.resource-group: Still des
57-...sourceGroups/rg-hawordpress-westeu-001, 10s elapsed]
module.resource_group.azure_rm_resource_group.resource-group: Destructi
Destroy complete! Resources: 23 destroyed.
maria@T14-PF3F4M9X:~/azure_ha_wordpress/Scripts/Terraform$ |
```



8. Conclusión

A lo largo del proyecto, hemos constatado que el uso de Terraform aplicado a Azure resulta una solución altamente eficiente y efectiva para implementar y gestionar infraestructuras en la nube. Terraform ofrece una forma declarativa y flexible de desplegar y definir recursos en Azure de manera escalable y reproducible.

Al emplear Terraform, obtenemos una perspectiva basada en código para la infraestructura, lo que facilita tareas como la automatización y el control de versiones de esta, permitiendo por ejemplo uso de herramientas de versionado de código como Git. Esto conlleva a una colaboración más efectiva entre los equipos de desarrollo y operaciones, lo que a su vez garantiza una implementación y administración confiable y rápida de la infraestructura.

En cuanto a la integración entre Terraform y Azure, el uso de ambas herramientas une la flexibilidad, adaptabilidad y escalabilidad de Azure junto con las altas capacidades de control, gestión y declaración de infraestructura que aporta Terraform. Esto redundará en la creación de entornos de infraestructura sólidos, reproducibles, flexibles, escalables y de mantenimiento y desarrollo automatizados.

En resumen, combinar Terraform con Azure nos permite simplificar y acelerar los despliegues de infraestructuras alojadas en servicios de nube pública, garantizando consistencia, confiabilidad y flexibilidad en todo el proceso.

9. Bibliografía

Alta disponibilidad para Azure SQL Database e Instancia administrada de SQL. (marzo 2023). Microsoft Azure. <https://learn.microsoft.com/es-es/azure/azure-sql/database/high-availability-sla?view=azuresql&tabs=azure-powershell>

Amazon Web Service Pricing Calculator. (marzo 2023). Amazon Web Service. <https://calculator.aws/#/>

Azure Database for SQL Server 2019. (marzo 2023). Microsoft Azure. <https://azure.microsoft.com/es-es/products/virtual-machines/sql-server>

Azure Load Balancer. (marzo 2023). Microsoft Azure. <https://azure.microsoft.com/es-es/products/load-balancer/>

Azure Provider. (septiembre 2021). Hashicorp. <https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs>

Benefits of Terraform with Azure. (10 abril 2022). Microsoft Azure. <https://learn.microsoft.com/en-us/azure/developer/terraform/overview#benefits-of-terraform-with-azure>

Curso Docker. (marzo 2023). GitHub. https://github.com/josedom24/curso_docker_ies

GitHub Education. (marzo 2023). GitHub. <https://education.github.com/>

Google Cloud Pricing Calculator. (marzo 2023). <https://cloud.google.com/products/calculator?hl=es>

Instalación de Terraform en Windows con Bash. (marzo 2023). Microsoft Azure. <https://learn.microsoft.com/es-es/azure/developer/terraform/get-started-windows-bash?tabs=bash>

Microsoft Azure. (marzo 2023). Microsoft Azure Cloud. <https://azure.microsoft.com/es-es>

Microsoft Azure for Students. (marzo 2023). Microsoft Azure. <https://azure.microsoft.com/es-es/free/students/>

Microsoft Azure Pricing Calculator. (marzo 2023). Microsoft Azure. https://azure.microsoft.com/en-us/pricing/calculator/?ef_id=k_CjwKCAjwuqiiBhBtEiwATgvixJdEeH1X7KVKBQBe4i2-w2ei8dA7POtd9N2Ens1YaquTGQSusn1wlBoCC-UQAvD_BwE_k_&OCID=AIDcmm68ejnsa0_SEM_k_CjwKCAjwuqiiBhBtEiwATgvixJdEeH1X7KVKBQBe4i2-w2ei8dA7POtd9N2Ens1YaquTGQSusn1wlBoCC-UQAvD_BwE_k_&gclid=CjwKCAjwuqiiBhBtEiwATgvixJdEeH1X7KVKBQBe4i2-w2ei8dA7POtd9N2Ens1YaquTGQSusn1wlBoCC-UQAvD_BwE

Nodos de Azure. (enero 2021). Avalora. <https://www.avalora.com/actualidad/actualidad-general/tecnolog%C3%ADas/microsoft->

[azure-abre-dos-nodos-en-espa%C3%B1a-por-qu%C3%A9-tus-datos-van-a-estarm%C3%A1s-seguros-que-nunca/](#)

Terraform. (marzo 2023). Hashicorp. <https://www.terraform.io/>

What is Ansible? (15 abril 2021) Red Hat.

<https://www.redhat.com/es/technologies/management/ansible/what-is-ansible>

What is Docker? (marzo 2023). Docker. <https://www.docker.com/>