Logotipo, Esquemático

Descripción generada automáticamente



**Laboratorio**

TEORIA

TERRAFORM

**CONTROL DE VERSIONES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado por:** | Jonatan Stiven Gutierrez | **No. de Versión:** | 1.0.0 |
| **Revisado por:** |  | **Fecha de revisión:** |  |
| **Aprobado por:** |  | **Fecha de Aprobación:** |  |

**Historia de Modificaciones**

| **No. de Versión** | **Fecha de Versión** | **Autor** | **Revisado por** | **Aprobado por** | **Descripción** |
| --- | --- | --- | --- | --- | --- |
| 1.0.0 | 21/02/2024 | Jonatan Stiven Gutierrez |  |  | Documento Original |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Lista de distribución**

| **Para** | **Acción\*** | **Empresa** | **Firma/Medio de Entrega** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*\* Tipos de acción: Aprobar, Revisar, Informar, Archivar, Complementar, Asistir a junta, Otras (por favor especificar)*

Contenido

[INTRODUCCION 3](#_Toc54124794)

[PRERREQUISITOS 3](#_Toc1076216077)

[TERRAFORM: 3](#_Toc1449750151)

[INTRODUCCION A LAC: 5](#_Toc282450505)

[QUE ES HCL: 5](#_Toc1297730414)

[MULTIPLE PROVIDERS Y DEFINICION DE DRY: 7](#_Toc1102826236)

# INTRODUCCION

El siguiente documento proporciona una introducción detallada a Terraform y sus diferentes temas.

# PRERREQUISITOS

* Haber cumplido con todo lo previamente visto.

# TERRAFORM:

Documentación: <https://registry.terraform.io>

Terraform es una herramienta de infraestructura como código (IaC) que permite a los equipos de operaciones y desarrollo definir y gestionar la infraestructura de manera programática. Desarrollado por HashiCorp, Terraform simplifica el proceso de aprovisionamiento, configuración y gestión de recursos en la nube y en otros entornos de infraestructura.

Características Principales:

1. **Declarativo y Legible:** Terraform utiliza un enfoque declarativo para describir la infraestructura deseada, lo que permite a los usuarios definir recursos utilizando una sintaxis sencilla y legible.
2. **Soporte para Múltiples Proveedores:** Terraform es compatible con una amplia variedad de proveedores de nube, incluyendo AWS, Azure, Google Cloud Platform, y más. Esto permite a los equipos gestionar recursos en diferentes entornos de manera coherente.
3. **Estado de la Infraestructura:** Terraform mantiene un estado de la infraestructura gestionada, lo que facilita la realización de cambios incrementales y la gestión de recursos de manera eficiente.
4. **Modularidad y Reutilización:** Terraform fomenta la modularidad y la reutilización del código mediante el uso de módulos, variables y plantillas, lo que permite evitar la duplicación de configuraciones y promover la coherencia del código.

Beneficios:

1. **Agilidad:** Terraform permite aprovisionar y configurar recursos de manera rápida y eficiente, lo que acelera el ciclo de desarrollo y despliegue de aplicaciones.
2. **Consistencia:** Al definir la infraestructura como código, Terraform garantiza que los entornos de desarrollo, pruebas y producción sean coherentes y reproducibles.
3. **Escalabilidad:** Con su soporte para múltiples proveedores y su capacidad para gestionar infraestructuras de cualquier tamaño y complejidad, Terraform es altamente escalable.

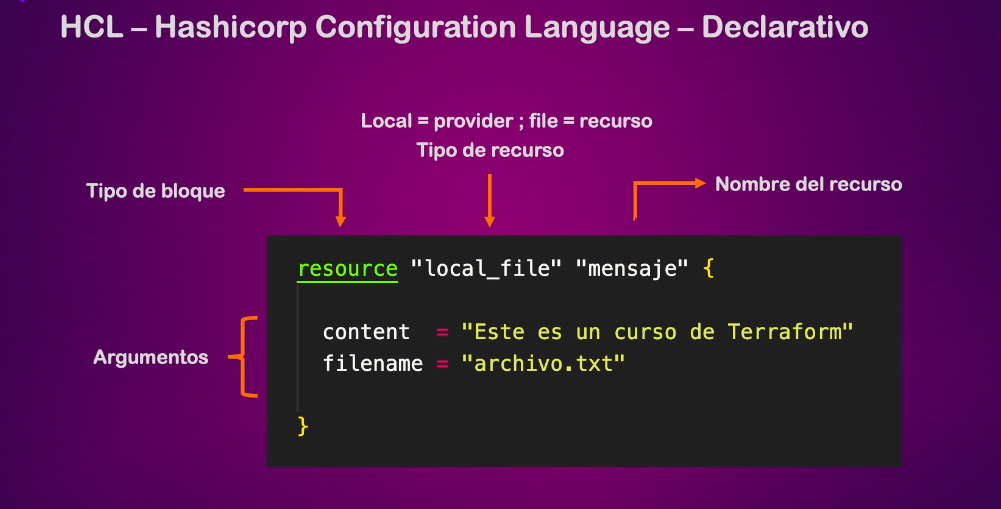
Terraform es una herramienta poderosa que simplifica la gestión de la infraestructura como código, proporcionando una forma eficiente y escalable de automatizar tareas de aprovisionamiento, configuración y gestión de recursos. Con su enfoque declarativo, soporte para múltiples proveedores y capacidad para seguir principios como DRY, Terraform es una opción popular para la automatización de la infraestructura en entornos de nube y de infraestructura.

# INTRODUCCION A LAC:

Terraform es una herramienta de infraestructura como código (IaC) desarrollada por HashiCorp que permite definir y gestionar la infraestructura de forma programática. Una de las características clave de Terraform es su lenguaje de configuración de alto nivel (HCL), que proporciona una sintaxis clara y expresiva para definir la infraestructura deseada.

# QUE ES HCL:

El Lenguaje de Configuración de Alto Nivel (HCL) es el lenguaje utilizado por Terraform para definir la infraestructura como código. HCL es un lenguaje simple y expresivo que permite a los usuarios describir recursos y configuraciones de manera fácil de entender y mantener. Proporciona una sintaxis clara y concisa para definir recursos, variables, bloques de configuración y más.

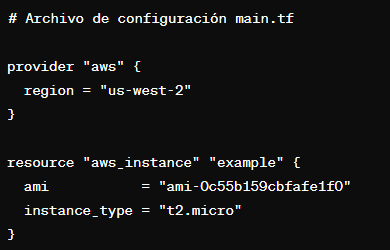


Características de HCL:

1. **Declarativo y Expresivo:** HCL utiliza un enfoque declarativo para definir la infraestructura deseada, lo que significa que describe qué recursos deben crearse y cómo deben configurarse, en lugar de especificar pasos detallados para crearlos. Esto hace que las configuraciones sean más legibles y fáciles de entender.
2. **Sintaxis Simple y Legible:** HCL utiliza una sintaxis simple y legible que facilita la escritura y comprensión de las configuraciones. Utiliza bloques de configuración y atributos clave-valor para definir recursos y configuraciones, lo que hace que las configuraciones sean fáciles de estructurar y organizar.
3. **Modularidad y Reutilización:** HCL es modular, lo que significa que puedes definir recursos y configuraciones en módulos separados y luego combinarlos para construir configuraciones más complejas. Esto promueve la reutilización del código y facilita la gestión de configuraciones grandes y complejas.

Ejemplo de Uso de HCL en Terraform:

Supongamos que queremos definir una infraestructura básica en AWS utilizando Terraform y HCL. Aquí hay un ejemplo simple que define una instancia de EC2:



En este ejemplo:

* Se define el proveedor de AWS y se especifica la región en la que se van a crear los recursos.
* Se define un recurso de instancia de EC2 llamado "example", especificando la AMI y el tipo de instancia.

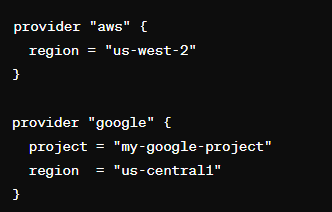
Este es solo un ejemplo básico, pero muestra cómo puedes utilizar HCL para definir recursos y configuraciones en Terraform. Con HCL, puedes crear configuraciones más complejas y gestionar infraestructuras de cualquier tamaño y complejidad de manera eficiente y escalable.

# MULTIPLE PROVIDERS Y DEFINICION DE DRY:

En Terraform, la capacidad de utilizar múltiples proveedores y aplicar el principio DRY (Don't Repeat Yourself) son características esenciales para la gestión eficiente de la infraestructura como código. A continuación, explicaremos qué significan y cómo se aplican en Terraform.

**Múltiples Proveedores:**

Terraform permite gestionar recursos en varios proveedores de nube, como AWS, Azure, Google Cloud Platform, y otros. Esto proporciona flexibilidad a los equipos de infraestructura para utilizar servicios de diferentes proveedores en un mismo proyecto. Para configurar múltiples proveedores en Terraform, simplemente se define cada proveedor dentro del bloque provider, especificando las credenciales y configuraciones necesarias para cada uno.

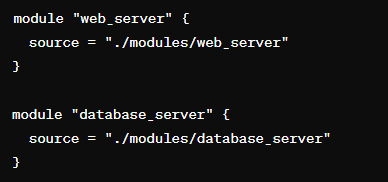


Con esta configuración, Terraform puede gestionar recursos tanto en AWS como en Google Cloud Platform en un mismo archivo de configuración.

**Principio DRY (Don't Repeat Yourself):**

El principio DRY es un concepto fundamental en la programación que enfatiza la importancia de la reutilización del código. En Terraform, aplicar el principio DRY implica evitar la duplicación de configuraciones y promover la modularidad del código. Esto se puede lograr utilizando módulos, variables y plantillas para definir recursos y configuraciones comunes una vez y reutilizarlas en múltiples partes del código.

Por ejemplo, en lugar de definir manualmente cada recurso de manera repetitiva, se pueden crear módulos que encapsulen conjuntos de recursos relacionados y luego utilizarlos en diferentes partes del código.



En este ejemplo, los módulos web\_server y database\_server encapsulan la lógica y la configuración necesarias para crear y gestionar servidores web y bases de datos, respectivamente. Luego, estos módulos pueden ser utilizados en diferentes partes de la infraestructura, evitando la duplicación de código y promoviendo la coherencia y la mantenibilidad.

En resumen, utilizar múltiples proveedores y aplicar el principio DRY en Terraform permite crear configuraciones de infraestructura más flexibles, eficientes y mantenibles. Esto facilita la gestión de la infraestructura como código en entornos de nube heterogéneos y complejos.

# DESPLIEGUE EN AWS:

...