
Universidad
Tecmilenio®

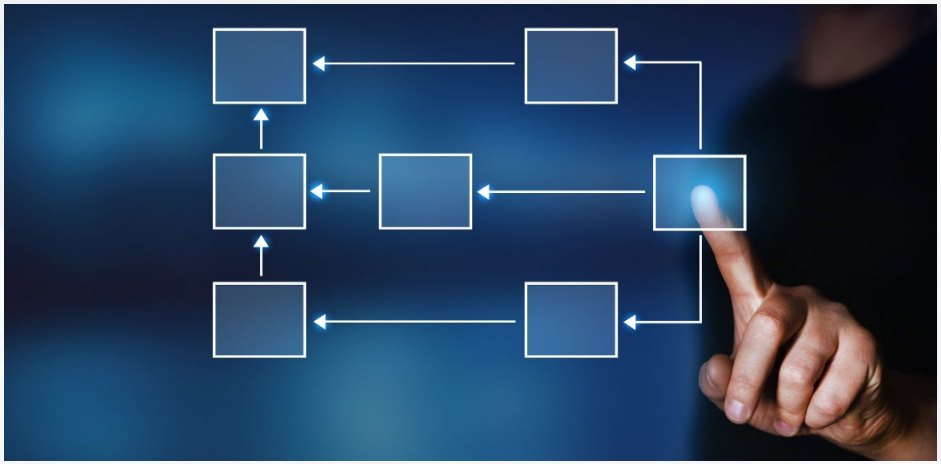
Universidad Tecmilenio

Infraestructura como código

Tipos de módulos en Terraform

Semana 6

Introducción



Al trabajar con proyectos de desarrollo de software es muy importante modularizar la solución, ya que así es posible dividir las cargas de trabajo para asignar el desarrollo, mantenimiento, pruebas y el despliegue de una manera más asertiva al separar las configuraciones

Explicación

Módulos locales

Un módulo está referenciado en el argumento **“source”** de un bloque **“module”**.

Estructura de archivos de un módulo: License, Readme.md ,main.tf ,variables.tf y outputs.tf.

Estructura de archivos de un módulo: License, Readme.md ,main.tf ,variables.tf y outputs.tf.

Readme.md: muestra el contenido describiendo el funcionamiento del módulo.

Main-.tf: contiene las configuraciones del módulo.

Variables.tf: contiene la definición de las variables del módulo.

Outputs.rf: contiene las definiciones de salida del módulo.

Explicación

Módulos remotos

Módulos remotos.	Una de las principales características del uso de los módulos es poder compartirlos, ya sea por colaboradores en la misma empresa u organización.	Para realizar el llamado a un módulo remoto, basta indicar en este nodo la ubicación de la carpeta contenedora de los archivos de configuración con extensión .tf.
Fuentes para almacenar y localizar módulos remotos.	Terraform Registry (repositorio oficial de Terraform) GitHub BitBucket	Git genéricos Repositorios Mercurial Direcciones HTTP Cubetas S3 Cubetas GCS

Explicación

Uso de salidas para pasar datos a otros módulos

Ejercicio

Paso 1.
Ingresa a la liga del módulo IamUser.

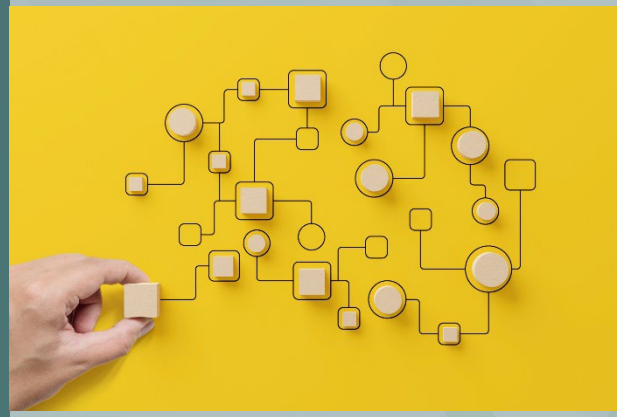
Paso 2.
Crea una carpeta de trabajo en la que crearás tres archivos.

Paso 3.
Agrega el argumento faltante en el archivo main.tf para que el módulo pueda ejecutarse adecuadamente y agrega el argumento de entrada que se requiere. Una vez hecho, ejecuta el comando “terraform init” y “terraform apply ”.

Paso 4.
Con base en los resultados, contesta las siguientes preguntas:
¿Qué tipo de bloque es el que se encuentra declarado en el main.tf?
¿De qué fuente se está obteniendo el módulo?
¿Cuántos argumentos son los requeridos?
¿Cuál es el argumento a ser especificado para crear en IamUser dentro de este módulo?
Al ejecutar correctamente el módulo, ¿cuáles son los outputs creados?

Paso 5.
Ejecuta terraform destroy para eliminar los cambios realizados.

Paso 6.
Elabora un documento a modo de reporte de proceso.de reporte de proceso.

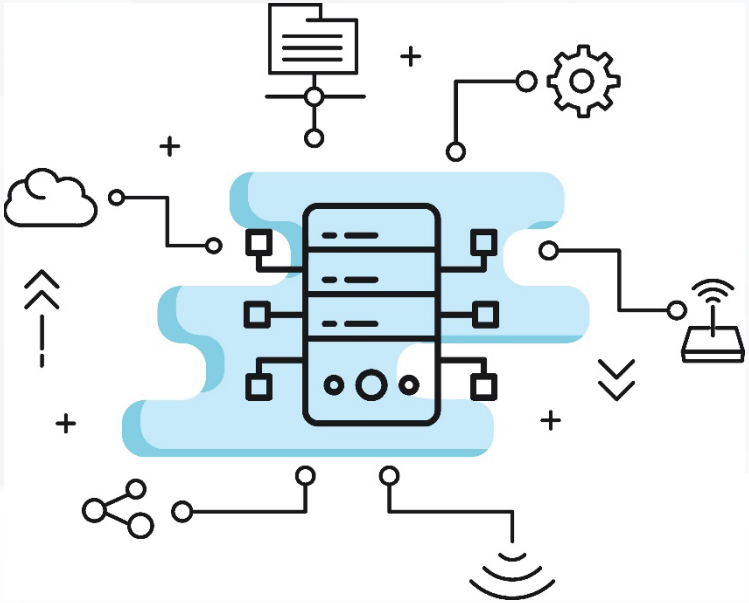


Cierre

Definir la infraestructura como código de Terraform utilizando módulos permite aplicar esta configuración a una gran variedad de proyectos de desarrollo.

Además, se considera una de las buenas prácticas en la ingeniería de software aplicada a DevOps, ya que le permite al desarrollador validar los cambios y actualizaciones de cada módulo a través de prácticas de revisión de código y pruebas automatizadas de manera independiente.

Esto también hace posible versionar los módulos y así contar con diferentes etapas de construcción de infraestructura o de pruebas.



Universidad Tecmilenio

Infraestructura como código

Usando Terraform para AWS

Semana 6

Introducción

Terraform es una herramienta *open source* desarrollada por Hashicorp que brinda el poder de automatizar y administrar infraestructura con servicios, utilizando un lenguaje declarativo y AWS (Amazon Web Services).

AWS es una plataforma que, de acuerdo con el mismo Amazon (s.f.), cuenta con más de 200 servicios tecnológicos en la nube.

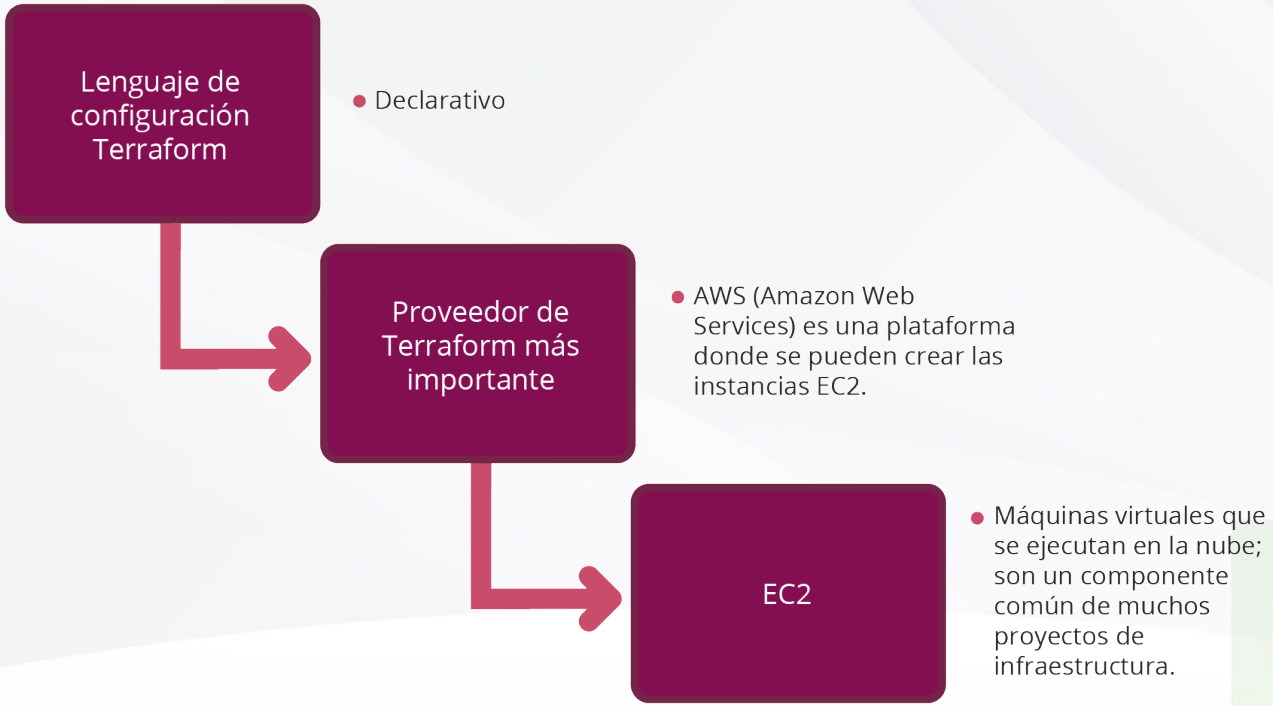
En conjunto, estas herramientas ofrecen una excelente alternativa para albergar proyectos de desarrollo de software de una manera eficaz y rápida.

Explicación

Usar Terraform para implementar infraestructura en AWS

Enfoque imperativo: se relaciona con las instrucciones de línea de comandos utilizadas en un cliente (CLI). El desarrollador verifica que la infraestructura funcione en algún proveedor (Yordanov, 2020).

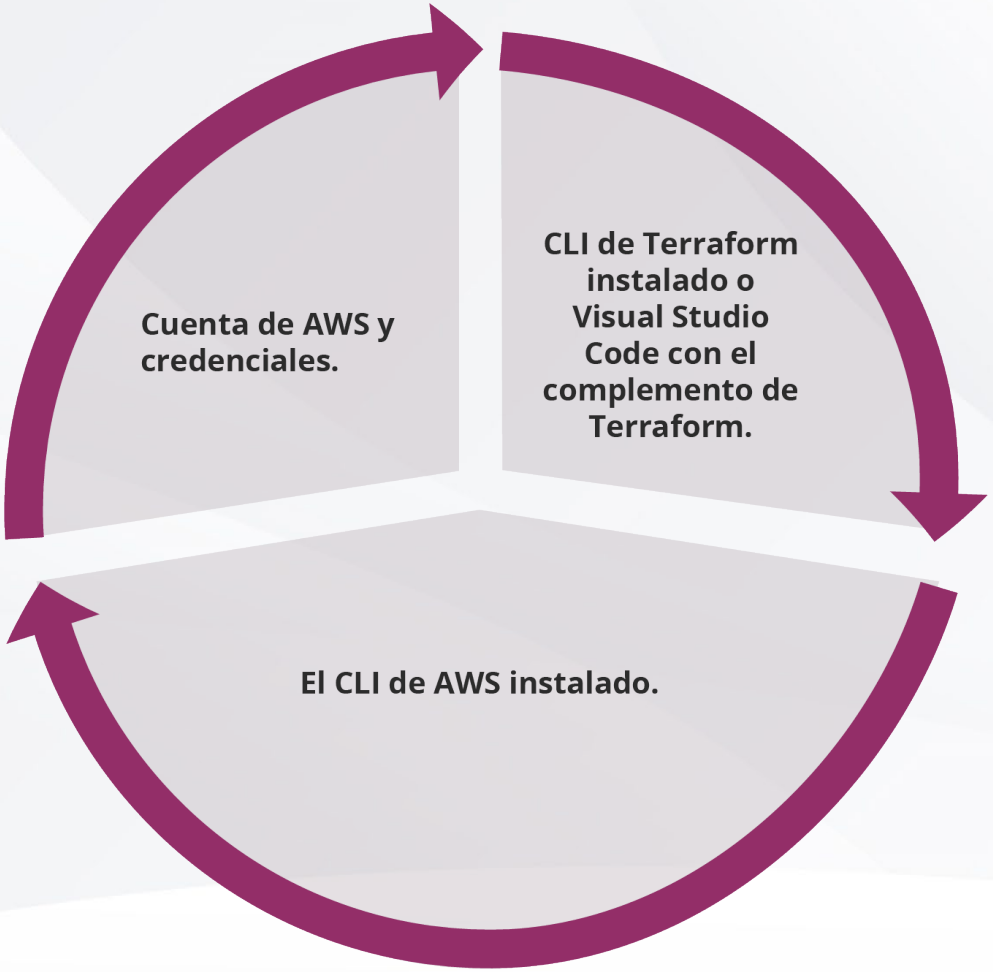
Enfoque declarativo: este enfoque define el estado final requerido de la infraestructura, sin ejecutar un solo comando y dejando al proveedor que se encargue del resto (Yordanov, 2020).



Explicación

Usar Terraform para implementar infraestructura en AWS

Elementos básicos para el uso de AWS:



Universidad Tecmilenio

Explicación

Usar Terraform para implementar infraestructura en AWS

Estructura de bloques:

Archivo con la información de Terraform	Incluye los proveedores necesarios que se utilizarán para aprovisionar la infraestructura. Se definen un nombre de host opcional, un espacio de nombres y un tipo de proveedor.
Proveedores (providers)	Configura los recursos de donde serán obtenidos, en este caso, los servicios de AWS.
Recursos (resources)	Un recurso puede ser un componente físico o virtual, como una instancia EC2, una cubeta S3, una base de datos DynamoDB o puede ser un recurso lógico, como una aplicación Heroku.

Explicación

Gestionar el estado de Terraform

Principales funcionalidades:

Universidad Tecmilenio

Explicación

Gestionar el estado de Terraform

Archivos de estados: el estado se almacena de forma predeterminada en un archivo local llamado “terraform.tfstate”, pero también se puede almacenar de forma remota. El comando terraform state es el que se usa para realizar modificaciones básicas del estado mediante la CLI.

Backends en Terraform: define en dónde se almacenan los estados de Terraform. Por defecto, Terraform tiene configurado un back end llamado “local”, cuya finalidad es guardar el estado en el disco. Todos los demás back ends respaldan el estado a través de un servicio remoto y generalmente requieren de datos de autenticación para salvaguardar la integridad y el acceso al estado.

Tipos de back ends: Local, Remote, Arifactory, Azurerm, Consul, Cos, Etcd, Gcs,Http, Kubernetes, Manta, Oss, Pg, S3 y Swift.

Configuración de back end: para habilitar un back end diferente al local, se deberá hacer uso del bloque terraform. Dentro de ese bloque se agregará el bloque back end especificando el tipo de back end a utilizar. Una configuración solo puede proporcionar un bloque “back end”.

Un bloque de “back end” no puede hacer referencia a valores con nombre (como variables de entrada, locales o atributos de fuente de datos), sin embargo, dependiendo del tipo de back end, se pueden utilizar sus atributos para inicializar datos de entrada.

Ejercicio

Paso 1.
Conoce el nombre del recurso, estructura y atributos de los servicios para:

- La autenticación en AWS.
- Creación de instancias EC2.
- Creación de un back end S3 para almacenar el estado remoto.

Paso 2.
Crea tu provider.

Paso 3.
Crea tu IaC para inicializar una instancia EC2 con Ubuntu LTS 16.04 en la zona us-west-1.

Paso 4.
Crea una cubeta S3 con la finalidad de almacenar el estado de manera remota de tu configuración de Terraform.

Paso 5.
Inicializa y aplica la configuración de tus módulos de Terraform.

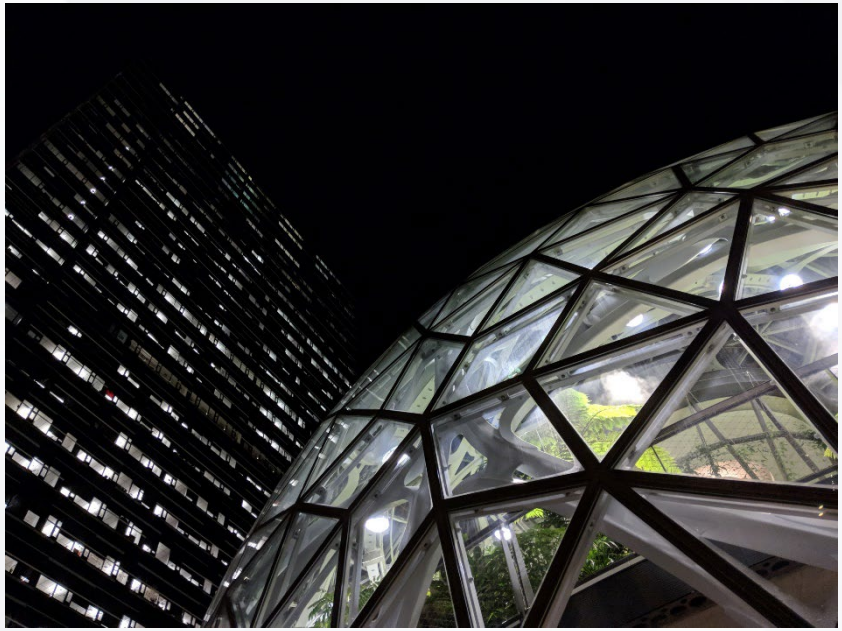
Paso 6.
Ejecuta los comandos “state list” y posteriormente “state show”.

Paso 7.
Destruye la configuración con el comando “terraform". destroy”.

¿Cuáles son los elementos que debes y necesitas considerar para que funcione?

¿AWS es la mejor opción o existen mejores alternativas? Si tu respuesta fue no, ¿cuáles serían?

Cierre



Actualmente, AWS es una plataforma muy popular para proyectos de desarrollo, pues los diferentes servicios que ofrece la hacen una de las más completas para generar infraestructura. Por ende, es una excelente opción para automatizar, modular y utilizar el aprovisionamiento empoderado que ofrece la tecnología IaC (infraestructura como código).

Bibliografía

•Amazon. (s.f.). *Informática en la nube con AWS.* Recuperado de https://aws.amazon.com/es/what-is-aws/

•Yordanov, V. (2020). *Primeros pasos con Terraform – crear instancia EC2 en AWS*. Recuperado de https://www.adictosaltrabajo.com/2020/06/19/primeros-pasos-con-terraform-crear-instancia-ec2-en-aws/.

Universidad Tecmilenio

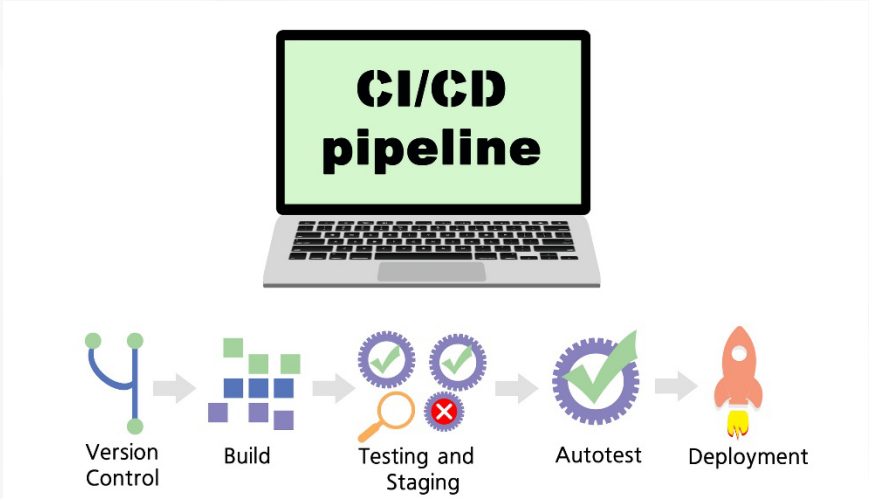
Infraestructura como código

Usando Terraform para CI/CD

Semana 6

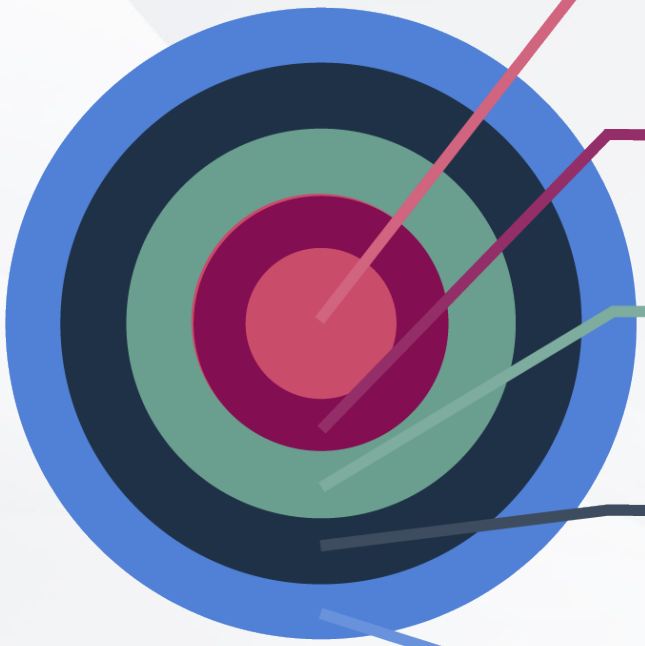
Introducción

Como ingeniero, sabes que existen muchas plataformas para aprovisionar infraestructura en la nube, como AWS, Azure, Docker, etc. Estos servicios se pueden hospedar en máquinas virtuales que ejecuten Linux o Windows, o microservicios alojados en contenedores, back ends de almacenamiento, etc. Independientemente de que se use Java, Node.js, Go, Flutter, PHP para codificar los proyectos de desarrollo de aplicaciones, es imperante usar correctamente las canalizaciones de integración continua e implementación (CI/CD), que permiten actualizar automáticamente los cambios, hacer pruebas más rápidas y precisas, trabajar en equipos mucho más eficiente, por mencionar algunas de las muchas ventajas de aplicar DevOps en los proyectos tecnológicos. Aquí es donde Terraform entra como un nuevo contendiente y potente aliado, ya que su tecnología de crear infraestructura como código permitirá la ejecución de las operaciones CI/CD con unos sencillos pasos.



Explicación

Terraform en un entorno CI/CD



Los equipos de trabajo que practican la cultura de CI/CD son más eficaces y ejecutan los despliegues de código mucho más rápido que quienes no lo hacen (Winkler, 2021).

El pilar fundamental del beneficio de utilizar Dev/Ops reside en que se mejora la calidad de los proyectos de desarrollo.

En un entorno de implementación y despliegue continuo Terraform funge como un orquestador, ya que determina mediante IaC (infraestructura como código), los pipelines CI/CD.

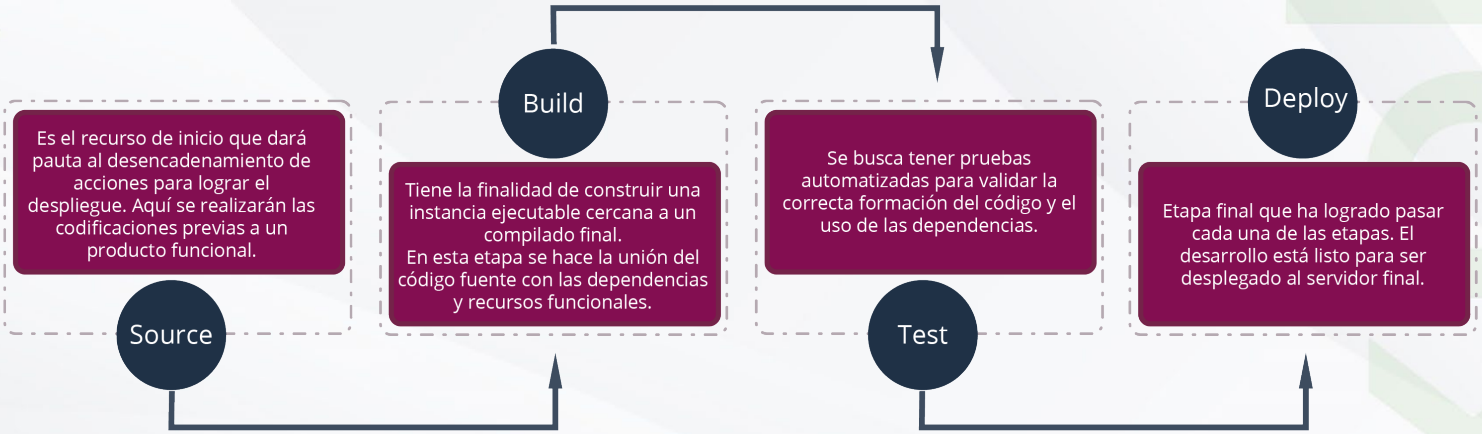
Los pipelines son los pasos para permitir entregar una nueva versión de software.

Al hablar de despliegues CI/CD, hay que tomar en cuenta que se facilita la gobernabilidad de los datos (secretos o información delicada). El objetivo de crear estos despliegues es facilitar y descentralizar todos estos flujos de datos con diferentes agentes para automatizarlo totalmente.

Explicación

Terraform en un entorno CI/CD

Cómo administrar el ciclo de vida de la infraestructura en un pipeline de CI/CD:



Explicación

Terraform en un entorno CI/CD

Elementos a considerar para ejecutar de forma correcta el acompañamiento del pipeline:

Almacenamiento de código.
¿Se deben almacenar en el mismo repositorio el código de Terraform (infraestructura) así como el del desarrollo (aplicación)? De acuerdo con Vasylenko (2021), no hay una respuesta correcta, ambos acercamientos son correctos, la decisión final dependerá del equipo de trabajo o del ingeniero Dev/Ops.

Preparar el entorno de ejecución de Terraform.
Aprovisionar el host para gestionar la ejecución de los pipelines y proveer control sobre el proceso; pueden usar contenedores en Docker, o bien, utilizar infraestructura en la nube que provee el entorno de ejecución de Terraform.

Automatizar el plan y el apply de cambios.
El comando "terraform apply" puede realizar ambos, pero puede ser útil separar estas acciones.

Presta atención a los entornos que no manejan el estado.
Solo en los casos que utilices los comandos "init", "plan" y "apply" deberás prestar atención a los archivos de Terraform que se pueden considerar como un artefacto.

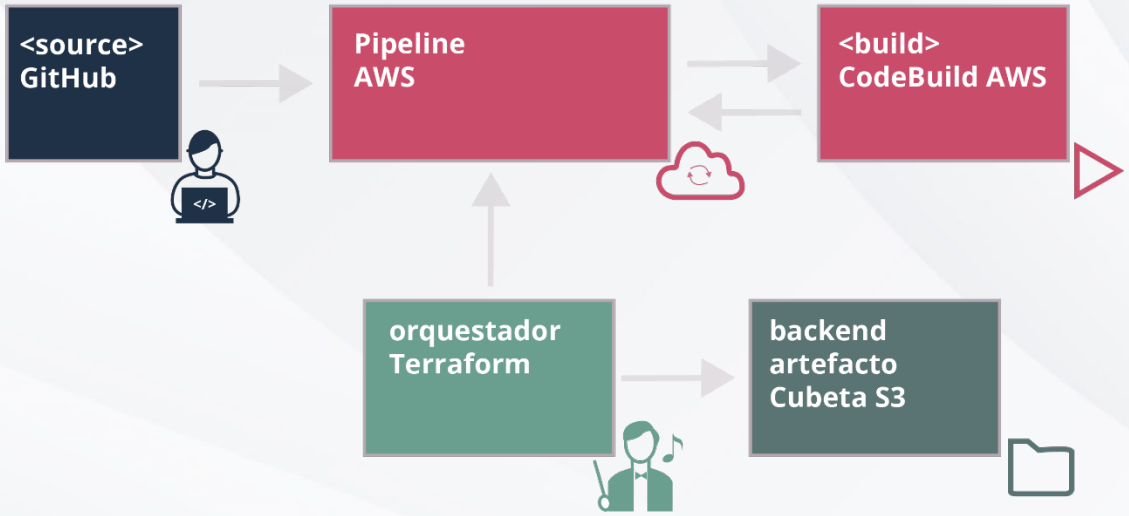
Considera el manejo correcto de las variables de entorno

Pueden ser configuradas para trabajar con los estados o trabajos de las diferentes herramientas CI/CD.

Explicación

Levantamiento de una infraestructura básica

Diagrama de orquestación de un CI/CD:



Para la canalización de pipelines, Amazon provee, dentro de las soluciones de su nube AWS, el recurso llamado AWS CodePipeline. Este recurso permite generar acciones a ejecutar en cada una de las etapas en el proceso de CI/CD. Las instrucciones y la configuración se pueden realizar desde el CLI de la plataforma, o bien, mediante archivos de configuración creados en formato yaml.

Ejercicio

Paso 1.
Realiza una conexión a un repositorio de GitHub y clona el repositorio.

Paso 2.
Analiza la estructura y los bloques de uso, configúralos con los datos de acceso de tu cuenta de GitHub.

Paso 3.
Inicializa la configuración (init).

Paso 4.
Aplica (apply) a la configuración y alimenta las entradas que haga la consola.

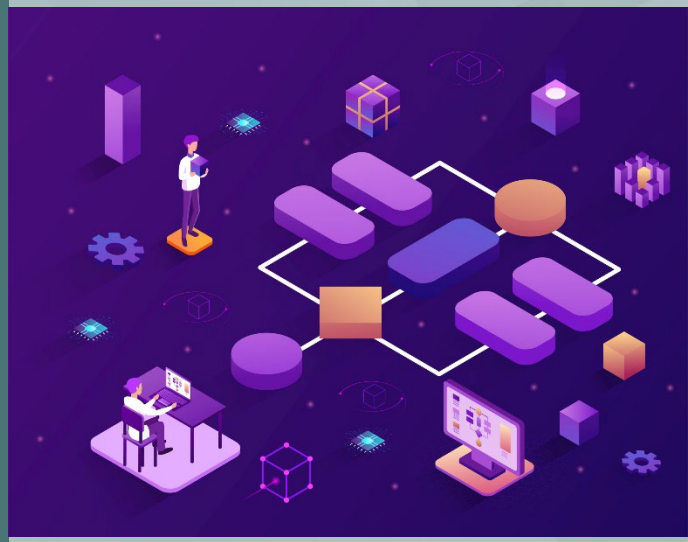
Paso 5.
Verifica tu cuenta de GitHub.

Paso 6.
Destruye (destroy) la infraestructura creada.

De acuerdo con las etapas del ciclo de vida de un pipeline:

¿Qué elementos necesitaste para realizar el despliegue de manera correcta?

¿Consideras que hay otra herramienta adicional a AWS que pueda hacer lo mismo y cuente con las mismas ventajas?



Cierre

Terraform será uno de los protagonistas en el futuro de la infraestructura como código. Con Terraform, los desarrolladores pueden aprovisionar recursos definidos por software en un flujo de trabajo automatizado, en lugar de depender de la configuración manual o de la configuración compartida en un repositorio compartido.

A medida que más desarrolladores adopten Terraform y lo incorporen a sus pipelines CI/CD, la gestión de la infraestructura será mucho más fácil y rápida.

Bibliografía

• Winkler S. (2021). *Terraform in Action*. Estados Unidos: Manning Publications Co.

• Vasylenko, S. (2021). *Guía para usar Terraform en CI/CD.* Recuperado de https://devdosvid.blog/2021/11/24/guide-to-using-terraform-in-ci/cd/

Universidad Tecmilenio

Gestión de la configuración

Gestión de la configuración de Ansible

Semana 6

Introducción

Ansible es una herramienta de automatización de código abierto diseñada para automatizar tareas de TI pasando instrucciones a sistemas de servidores remotos a través de SSH.

Ansible presenta retos para los usuarios, debido al gran número de comandos que hay que memorizar y la curva de aprendizaje puede llegar a ser demandante en los primeros pasos.

Sin embargo, al ser una de las primeras herramientas que permite gestionar todas las tareas, independientemente de la plataforma, permite que los desarrolladores se liberen, en gran medida, de la carga de trabajo, para que se concentren en proyectos estratégicos y no en la gestión de la infraestructura.

Explicación

Introducción a Ansible

Ansible automatiza los procesos para preparar la infraestructura, gestionar la configuración, implementar las aplicaciones y organizar los sistemas.

Cifrado asimétrico: este método utiliza un par de claves para el cifrado y descifrado, que se conocen como claves públicas y privadas. La clave pública, como sugiere su nombre, se distribuye ampliamente. La clave privada está estrechamente relacionada con la clave pública en términos de funcionalidad, pero no se puede calcular simplemente conociendo la clave pública.

Asible tiene el propósito de conectarse a los nodos (host) e insertar fragmentos de código, los cuales reciben el nombre de módulos, estos realizan tareas de automatización en la plataforma, lo que permite ejecutar dichos programas que funcionan como modelos de recursos del estado deseado de los sistemas y los quita cuando finaliza (Red Hat, 2020).

Utilizar el protocolo SSH conlleva usar claves y un cliente SSH de forma predeterminada para conectarse a las máquinas remotas con el nombre de usuario actual.

Cifrado simétrico: también conocido como cifrado de clave compartida, suele ser una sola clave o un par de claves que se utilizan tanto para el cifrado como para el descifrado de un mensaje. Esta clave se usa para cifrar toda la sesión de comunicación entre un cliente y un servidor.

Explicación

Herramientas para la gestión de la configuración

Es un proceso que busca mantener los sistemas informáticos, servidores y el software en un estado óptimo y uniforme para garantizar que el sistema se pueda ejecutar de manera correcta a medida que se realizan cambios.

Algunas herramientas para la gestión de la configuración: Ansible, CF Engine, Chef, Puppet y Salt.

Explicación

Diferencia entre IaC y herramientas de gestión de configuración

Características	Terraforma (IaC)	Ansible
Permite la orquestación de servicios en infraestructura.	Sí	No
Permite el proceso de automatizar la instalación de componentes de software versionado.	Sí	Sí
Manejo del estado.	Sí	No
Libre de agentes.	Sí	Sí
Aprovisionamiento de infraestructura.	Sí	No
Administración del ciclo de vida.	Sí	No
Infraestructura inmutable.	Sí	No
Enfoque declarativo.	Sí	No

Ejercicio

Paso 1.
Revisa los detalles de configuraciones y requerimientos de instalación para las diferentes plataformas.

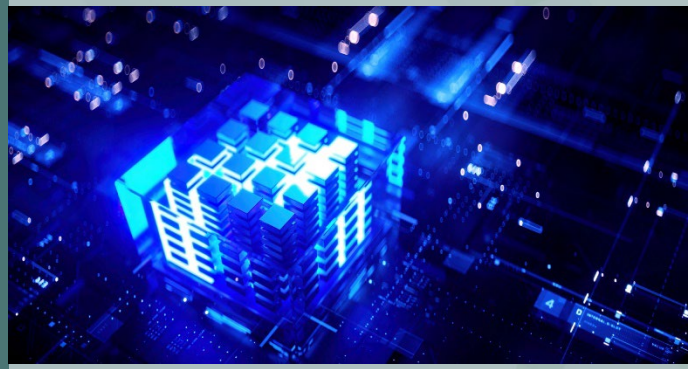
Paso 2.
Prueba si la instalación se realizó de manera correcta y para ello se configurará un archivo de inventario.

Paso 3.
Realiza una prueba de conexión.

Paso 4.
Documenta la instalación.

¿Cuál herramienta consideras que es mejor?, ¿por qué?

¿Cuáles serían los elementos que determinarían elegir el software adecuado?, ¿por qué?



Cierre

La gestión de la configuración es el proceso de seguimiento, manejo y actualización de las configuraciones. El proceso garantiza que todos los equipos tengan las cargas de software y los datos correctos. Ansible es una alternativa, a diferencia de las técnicas tradicionales de gestión de la configuración, que permite gestionar miles de máquinas simultáneamente, copiar automáticamente las configuraciones y supervisar continuamente los cambios. Sin embargo, Ansible no es perfecto, ya en el aprovisionamiento de infraestructura por sí solo no es su fuerte, por ello, se sugiere complementarlo y trabajar en conjunto con otras tecnologías como Terraform (IaC).

Bibliografía

• Red Hat. (2020). *Conceptos básicos de Ansible*. Recuperado de https://www.redhat.com/es/topics/automation/learning-ansible-tutorial