C1 A: Ejes de la Materia Infra II.

Durante la cursada vamos a estar trabajando a partir de **tres grandes temas** o ejes que organizan la materia: **Infraestructura como código, Pipelines de CI/CD y Monitoreo**.

**Kahoot:**

1- El eje principal de la materia es infraestructura como código: Falso

2- El enfoque de infraestructura como código combina prácticas de: Desarrollo e infraestructura

3- La implementación de pipelines se lo conoce como “CI/CD”: Verdadero

4- Docker es una herramienta que nos permite: Construir nuestras aplicaciones en un formato portable.

C2 A: Introducción a DevOps.

**DevOps:** es la combinación de filosofías, prácticas y herramientas que incrementan la  
velocidad a la que una organización entrega aplicaciones y servicios, permitiendo mejorar los productos a un ritmo más rápido que las organizaciones que usen procesos  
de desarrollo e infraestructura tradicionales. Esta velocidad le permite a las organizaciones entregar más valor a sus clientes y ser más competitivas en el mercado.

* En la cultura DevOps, los equipos de desarrollo (Dev) y operaciones (Ops) no trabajan por separado, sino que se comunican de forma constante
* Estos equipos hacen foco en automatizar procesos que históricamente fueron manuales y lentos.

**DevOps; Automatización y monitoreo.**



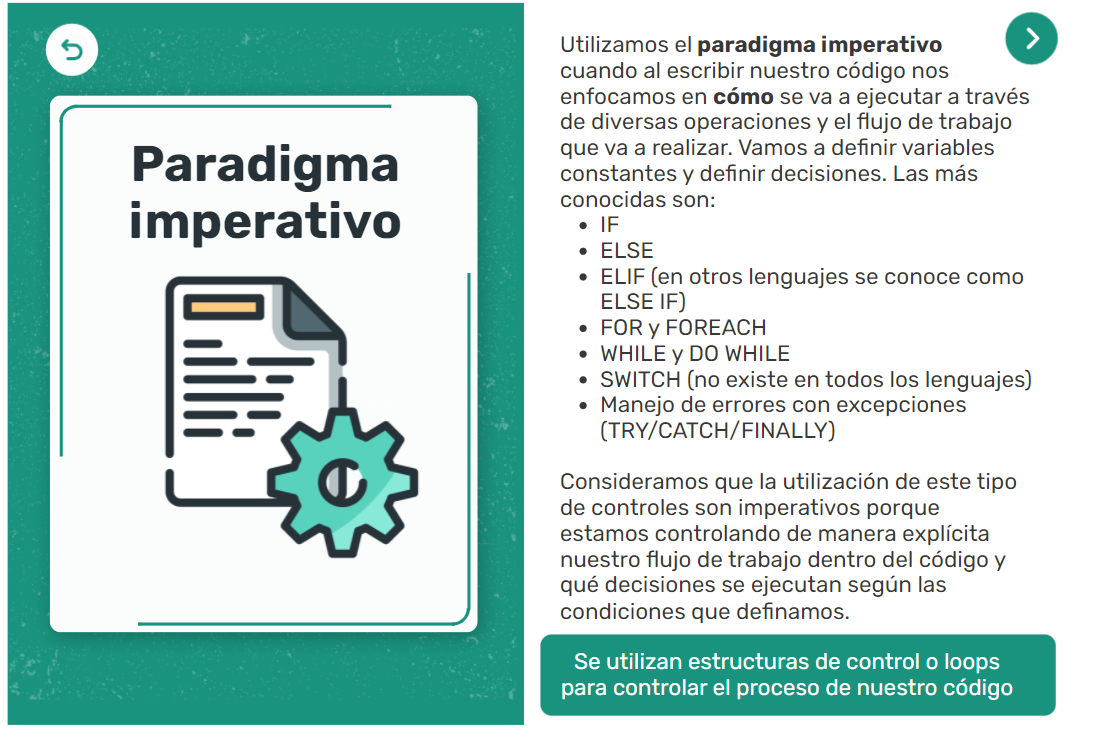
Roles/Perfiles en el ecosistema DevOps:

1. **Desarrollador de Aplicaciones:** Son quienes desarrollan la aplicación, los programadores front-end, back-end, mobile, full stack o especializados en una tecnología particular
2. **Analistas de calidad (QA):** Son quienes verifican y validan la aplicación. En un entorno DevOps es importante que también se concentren en automatizar pruebas para hacerlas repetibles y confiables.
3. **Analistas de Infraestructura:** Son quienes implementan la infraestructura sobre la cual se ejecutarán las aplicaciones y las bases de datos.
4. **Analistas de redes:** Se ocupan de la interconexión entre distintos sistemas, es decir, de las redes de computadoras —sean físicas o virtuales—.
5. **Analistas de seguridad:** Son personas que trabajan en la seguridad de la aplicación y de la infraestructura.
6. **Analistas de CI/CD:** Son quienes mantienen los pipelines de integración y despliegue continuos.
7. **Arquitecto de nube:** Definen la arquitectura del entorno en la nube: la estructura que tendrán los servidores, cómo se interconectan y varios aspectos de seguridad relacionados.
8. **Ingenieros de confiabilidad de sitio (sre):** Son los encargados de diseñar y monitorear el sistema para minimizar las suspensiones de servicio y el tiempo de recuperación de los servicios.
9. **Gerente de entregas:**  se ocupa de coordinar la entrega de nuevas versiones del producto a los clientes, llevar registro de qué cliente tiene qué versión del producto y orientar los esfuerzos del equipo hacia la satisfacción de los clientes.

C4 A: Infraestructura como código: La disciplina

**Infraestructura como código (IAC):**  Es la gestión de la infraestructura a través de templates que tienen la capacidad de ser versionados. De esta forma vamos a poder automatizar los procesos manuales que se requieren para lograr el objetivo final que buscamos.  
Así como cada vez que ejecutamos el código de nuestra aplicación obtenemos el mismo resultado, lo mismo ocurre con la infraestructura: vamos a obtener el mismo resultado de infraestructura desde nuestra IaC.

Existen dos paradigmas de programación aplicados a la Infraestructura como código. Al escribir nuestro IaC podemos optar por el **paradigma imperativo**, que nos posibilita controlar el flujo de trabajo de nuestro código, o bien enfocarnos en el resultado final y en el cambio de nuestra infraestructura, el **paradigma declarativo**. Es el "cómo" versus el "qué".





**El principio de idempotencia:** La idempotencia es un principio matemático utilizado en infraestructura. Pero, ¿qué es? Podemos definir la idempotencia como la propiedad de utilizar una automatización “n” cantidad de veces obteniendo siempre y en todos los casos el mismo resultado.

**Herramientas de IAC:**

**Terraform:** Es un software de código libre desarrollado por HashiCrop. Es una herramienta declarativa de aprovisionamiento y orquestación de infraestructura que permite automatizar el aprovisionamiento de todos los aspectos de la infraestructura, tanto para la nube como la infraestructura on-premise (en los mismos datacenter). Tiene algunas características interesantes, como comprobar el estado de la infraestructura antes de aplicar los cambios. Es la herramienta más popular porque es compatible con todos los proveedores de nube sin realizar modificaciones en nuestros templates.

**AWS CloudFormation:** Es la solución nativa de AWS para aprovisionar recursos en esta nube. En este caso se pueden definir templates en formato JSON o YAML. Se pueden utilizar para crear, actualizar y eliminar recursos las veces que sea necesario. Una ventaja de CloudFormation es que, al ser un servicio propio de Amazon, tiene una integración completa con los demás servicios de AWS, por lo que es nuestra mejor opción si solo utilizamos este proveedor de nube.

**Azure Resource Manager:** Es la herramienta nativa en Azure para implementar infraestructura como código, Azure Resource Manager (ARM Templates). Estas plantillas llevan una sintaxis declarativa en formato JSON, que nos permiten definir los recursos y las propiedades que conforman la infraestructura.

**Google Cloud Deployment Manager:** Google Cloud Deployment Manager es la herramienta IaC para la plataforma Google Cloud —lo mismo que CloudFormation es para AWS—. Con esta herramienta los usuarios de Google pueden administrar fácilmente mediante archivos de configuración YAML.

**Ansible:** Es una herramienta de automatización de infraestructuras creada por Red Hat. Ansible modela nuestra infraestructura describiendo cómo se relacionan sus componentes y el sistema entre sí, en lugar de gestionar los sistemas de forma independiente.

C5 A: Infraestructura como código en AWS CloudFormation

CloudFormation crea y configura la infraestructura que definimos previamente en una plantilla (o template) de acuerdo a los requisitos que necesitamos. Esto nos ofrece algunas ventajas, como crear repositorios con nuestros templates para que sean accesibles o que se puedan realizar entregas rápidas de los recursos de infraestructura.

Como usarla: Se necesitan:

* **Plantillas o templates:** es un archivo de texto con formato JSON (JavaScript Object Notation) que describe los recursos que queremos crear junto a sus propiedades.
* **Pilas:** es una unidad que genera CloudFormation para la creación ordenada de los recursos.
* **Cambios:** es un resumen de los cambios que se proponen para anticiparnos al resultado final.



Cuestionario PG:

1- ¿Qué herramienta se utiliza para versionar los templates de CloudFormation?  AWS que nos permite almacenar archivos estáticos y versionar los cambios.

2- ¿Cuál de estas extensiones es alguna de las válidas para escribir los templates de CloudFormation? Yaml es una de las extensiones válidas, como así también lo es JSON.

3- ¿Qué paradigma de programación podemos elegir para escribir la lógica de nuestros templates? El paradigma declarativo lo utilizamos en IaC para desarrollar templates compuestos por funciones, haciendo foco en el resultado de nuestra infraestructura.

CloudFormation se puede utilizar en distintos ámbitos:

* Podemos hacerlo por línea de comando desde nuestros equipos.
* En scripts (como PowerShell).
* En pipelines, como parte de un conjunto de tareas automatizadas y encadenadas entre sí, formando una tubería con un inicio y un fin.



C7 A: Infraestructura como código: Ansible

**Ansible:** Ansible es un proyecto comunitario *open source* diseñado para ayudar a las organizaciones a automatizar el aprovisionamiento de infraestructura, la gestión de configuración y el despliegue de aplicaciones.

Con Ansible se crean archivos de configuración llamados *playbooks*, escritos en YAML, que se utilizan para especificar el estado requerido de la infraestructura. Al ejecutarlos, Ansible se ocupa de aprovisionar la infraestructura necesaria para alcanzar el estado descrito.

**Como usarla:**

* No necesita instalación de agentes.
* Su configuración es de fácil lectura.
* Es muy flexible (usa APIs y *plugins*).
* Es fácil de usar por basarse en YAML.

En el siguiente video recorrerás la herramienta y verás cómo ejecutar tus propios *playbooks***.(PG)**

El corazón de Ansible es la ejecución de archivos de *playbooks* con las instrucciones necesarias para lograr la infraestructura que queremos, para no tener que escribir la lógica repetidamente para cada caso de uso, y que nuestro código utilice el paradigma declarativo.



Preguntas Ansible:

1. ¿Qué estrategia poseemos para reutilizar nuestro propio código?  Los roles nos permiten reutilizar la lógica para un problema similar al que nos enfrentamos.

2. ¿Qué tipo de agente es necesario instalar en los servidores de destino para administrar nuestra infraestructura?  No es necesario instalar ningún tipo de agente para utilizar Ansible.

3. ¿Qué protocolos utiliza Ansible para conectarse en servidores Linux y Windows? Ansible utiliza el protocolo SSH para conectarse a servidores Linux y WinRM para servidores Windows. Antes de usarlos no olvides configurar tus credenciales.

La flexibilidad de Ansible permite hacerlo en diferentes espacios:

* En tu computadora.
* En un servidor que pueda ser usado para ejecutar Ansible.
* En el proyecto de código abierto AWX que podés instalar y usar para administrar tus playbooks.



C8A: Infraestructura como código: Terraform

**Terraform:** es una herramienta de código abierto desarrollada por HashiCorp. Esta herramienta te permite definir y aprovisionar la infraestructura completa utilizando un lenguaje declarativo que podés ejecutar como infraestructura como código.

Si bien es similar a herramientas como CloudFormation, hay una gran diferencia: no es solo para AWS, sino que podés utilizarla con el resto de los proveedores de infraestructura cloud.

Al ser declarativo te permite escribir tu código en el lenguaje de alto nivel HCL (HashiCorp Configuration Language) para describir el estado final que deseas de tu infraestructura.

Actividad PG:

1. Si quiero ejecutar un template de Terraform lo hago con el comando...

2. ¿Cuál es la palabra reservada para hacer referencia a un proveedor de infraestructura dentro nuestros templates?

3. ¿Por qué decimos que Terraform es código declarativo?

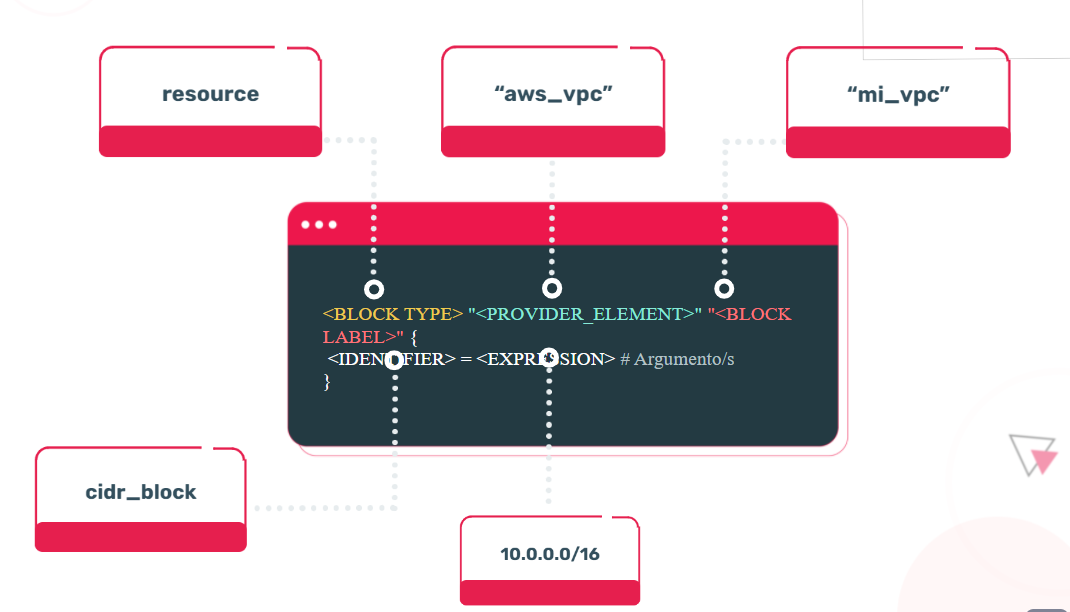




C10A: Infraestructura como código: Terraform (Continuación)

**Los plugins de Terraform:** providers y provisioners => PDF Clase 10 - Detrás de escena

**Domain Specific Language:** HCL es el lenguaje de configuración estructurado que nos permite interactuar con los servicios de infraestructura cloud.

****

**Resourse:** Este bloque iniciará el diálogo con nuestro proveedor cloud. Específicamente estamos diciendo: todo lo que continúa y con lo que voy a trabajar es con un recurso.

**Aws\_vpc:** Acá estamos declarando el recurso específico de AWS a utilizar. En este caso, un VPC.

**Mi\_vcp:** ¿Cómo lo voy a llamar? ¿De qué forma voy a referenciar si quiero invocar este elemento desde otro módulo?

**Cidir\_block:** Un valor definido por HCL. Depende del recurso que declaramos. En este caso, de “aws\_vpc”.

**10.0.0.0/16:** Representa un valor asociado al identificador. Por ejemplo: región = “us-east-1”, donde “us-east-1” es el valor.

Terraform HCL - Hashicorp Custom Language (GUIA PDF)

Video paso a paso en PG: Levantar una red publica y una privada y sintaxis de Terraform.

<https://playground.digitalhouse.com/course/e42963b6-7132-4e0f-b98a-1690c96d3de3/unit/8bcf7f6a-25d1-4fff-ab15-f2801a10b906/lesson/8149c963-92b9-4ff5-813c-cf45d5e84a22/topic/f401c12e-b8f8-4514-9a35-1cb5cb8269e0>

CloudFormation vs Anisble vs Terraform : Video “Bonus track”