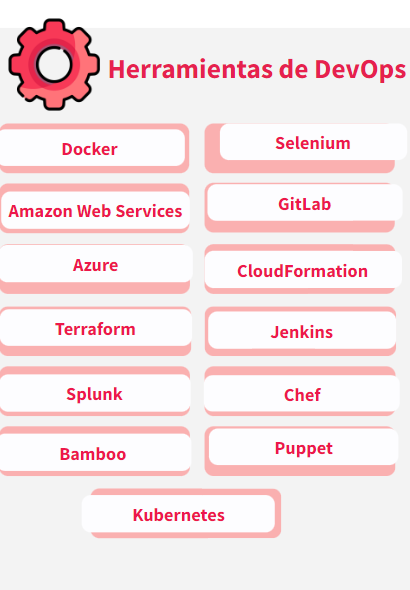
Infraestructura II

MODULO 1

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. A veces los equipos se fusionan en un único equipo que trabaja en el ciclo de vida completo de la aplicación, desde el desarrollo y las pruebas hasta el despliegue y la operación. En algunos casos, la calidad y la seguridad también se integran con el desarrollo y las operaciones.

Estos equipos hacen foco en automatizar procesos que históricamente fueron manuales y lentos. Para eso, utilizan un stack de tecnologías y herramientas que los ayuda a operar y evolucionar aplicaciones de forma rápida y confiable. Además, colaboran a que una sola persona pueda realizar tareas que normalmente requerirían de una combinación de varias personas, como desplegar código o aprovisionar infraestructura.



PERFILES DevOps:

* Desarrollador
* Analista de calidad
* Analista de infraestructura
* Analista de redes
* Analista de seguridad
* Analista de CI/CD
* Arquitecto de nube
* Ingeniero de confiabilidad de sitio (SRE)
* Gerente de entregas

EXPECTATIVAS DevOps:

* Automatizar
* Medir y monitorear
* Reducir la fricción
* Aumentar la comunicación
* Ser dueños de la solución

CI/CD:

Integración continua(CI): Integrar a main frecuentemente.

Despliegue continuo(CD): Desplegar a producción frecuentemente.

MONITOREO:

Monitorear servidores: Disponibilidad, carga.

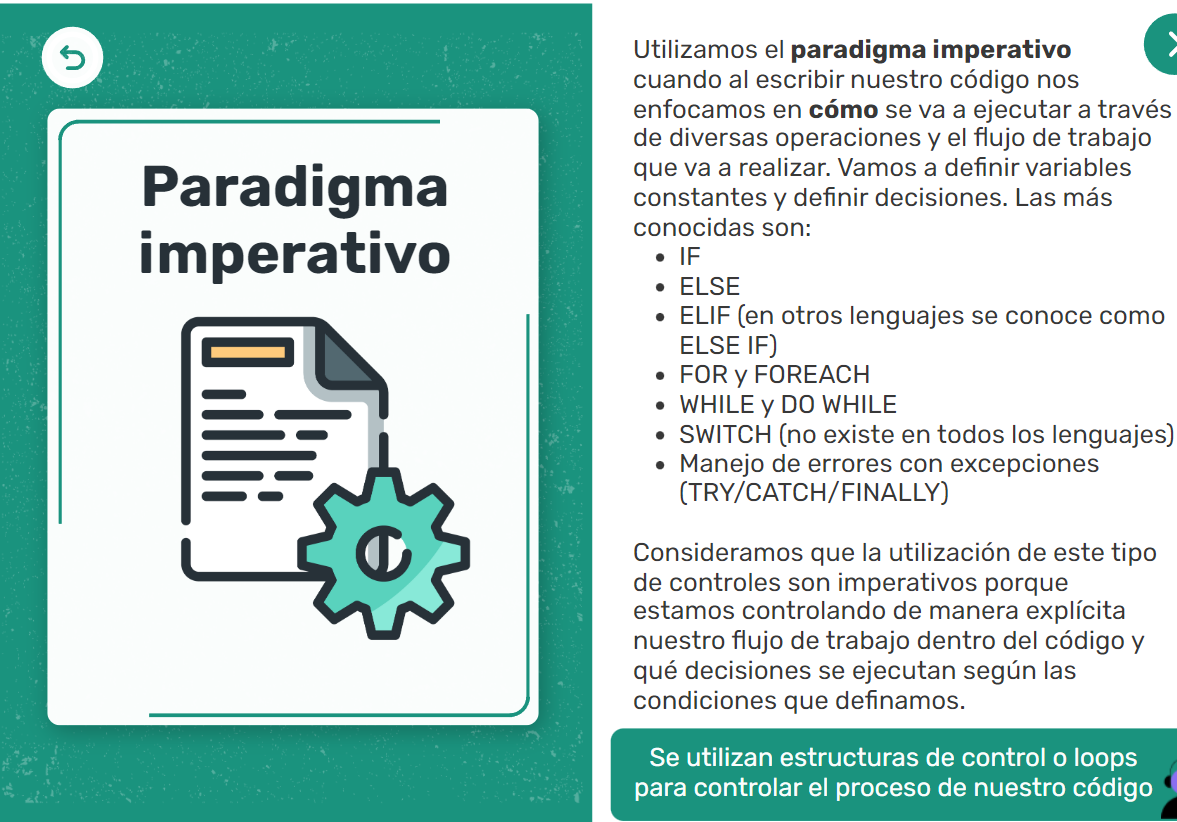
Monitorear aplicaciones: Performance, experiencia de usuario.

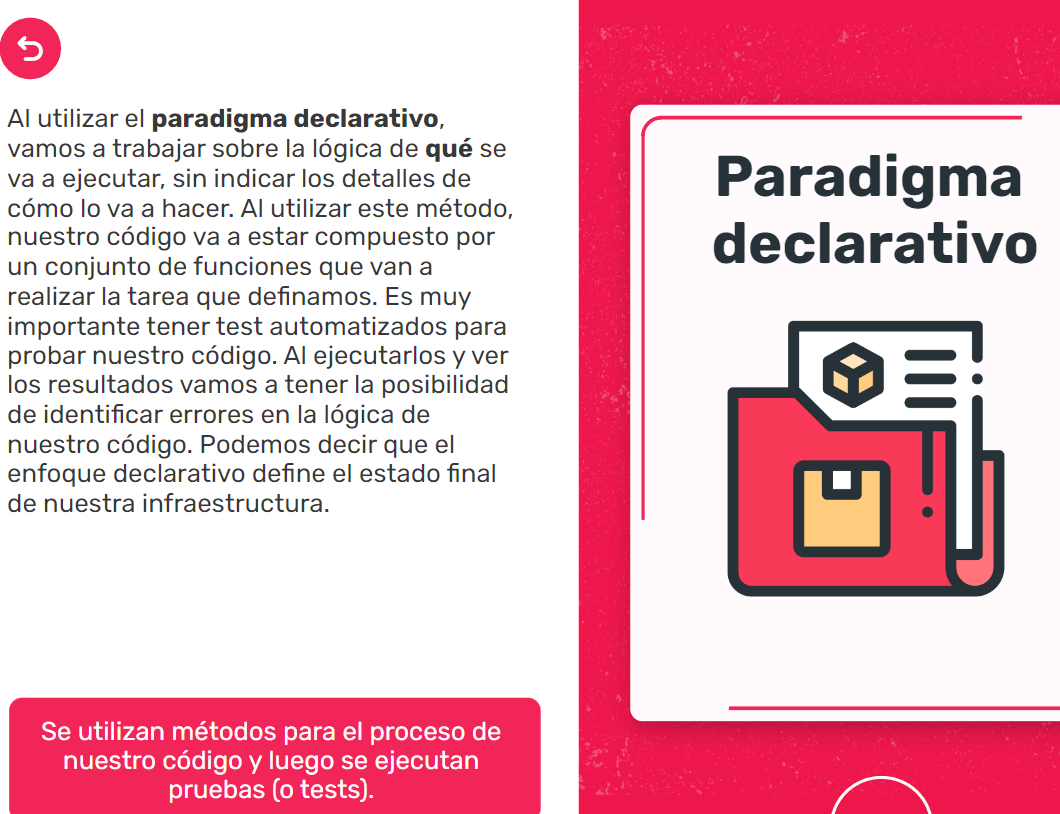
MODULO 2

INFRAESTRUCTURA COMO CODIGO (IaC): 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.

Al momento de realizar los análisis para implementar una infraestructura para nuestra aplicación, lo primero a definir es la arquitectura que necesitamos: qué servidor es el adecuado o qué base de datos necesitamos. Una vez seleccionado el tipo de servidor, avanzamos en la configuración e instalación de aquello que nuestro sistema operativo requiere para estar operativo.

PARADIGMAS DE PROGRAMACION APLICADOS A LA INFRA COMO CODIGO





ECOSISTEMA DE HERRAMIENTAS IaC

**TERRAFORM**

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**

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**

ARM 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.

**ANSIBLEANSIBLE**

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.

INFRAESTRUCTURA COMO CODIGO 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.

CONCEPTOS IMPORTANTES:

-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.



LINKS UTILES PARA CLOUDFORMATION:

<https://docs.aws.amazon.com/es_es/AWSCloudFormation/latest/UserGuide/sample-templates-applications-us-west-1.html>

<https://s3.us-west-1.amazonaws.com/cloudformation-templates-us-west-1/WordPress_Single_Instance.template>

SEMANA 3:

ANSIBLE (HERRAMIENTA DE INFRAESTRUCTURA COMO CODIGO):

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. Asimismo, es importante tener en cuenta que es fácil de aprender.

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.

Esto quiere decir que se puede, por ejemplo, crear una máquina virtual en el proveedor de infraestructura -como una instancia EC2 dentro de AWS- aplicando metodologías de infraestructura como código.

¿Como se usa?

Ansible es una herramienta que permite gestionar las configuraciones de tu infraestructura. Sus principales ventajas al momento de usarla son:

- 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.



¿dónde se pueden ejecutar?

- 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.

TERRAFORM

Terraform es una una herramienta de código abierto desarrollada por HashiCorp y la última de las tres que vas a conocer en las clases de Infraestructura como código.

Esta herramienta te permite definir y aprovisionar la infraestructura completa utilizando un lenguaje declarativo que podés ejecutar como infraestructura como código.

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.





Documentación: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs>

MODULO 3

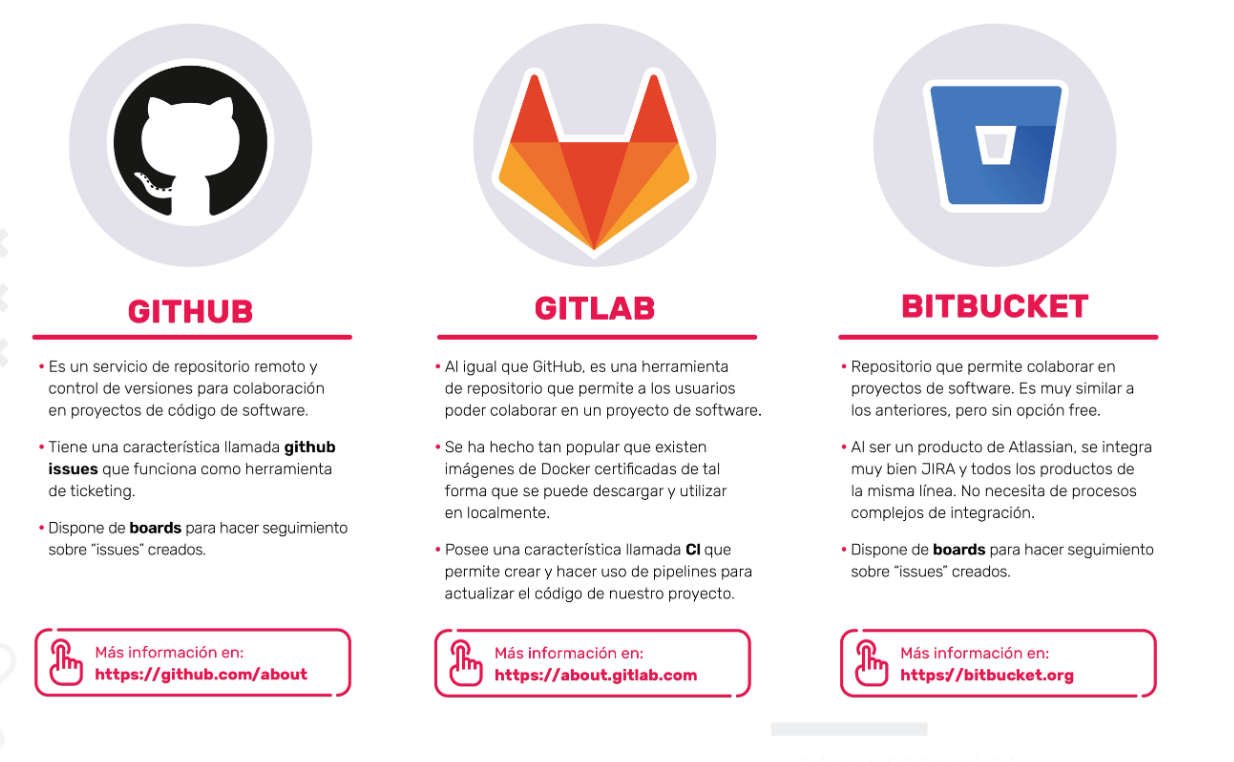
Pipelines es una práctica que nos sirve para agilizar los procesos de desarrollo e implementación de software que se vale de automatizar pasos repetitivos según  
un procedimiento preestablecido. Esta arquitectura es bastante común en el desarrollo de software porque tiene la propiedad de agilizar los procesos de build, deploy y release.

Un pipeline no es un proceso monolítico, sino que puede consistir en una o varias etapas que están encadenadas o secuenciadas y que van transformando a nuestro elemento inicial de acuerdo con ciertas reglas hasta llegar al producto final.

¿Qué tecnologías existen?

Todos estos productos disponen de flujos de trabajo altamente configurables según las necesidades del usuario. Pero, ¿qué son los flujos de trabajo? ¿De qué se tratan? Los flujos de trabajo dan cuenta de las características innatas o facilidades que estos sistemas nos ofrecen:

- Almacenamiento de código, Versionado, CI/CD, Control de cambios, Colaboración y revisión



Continuous integration

Pensemos en una aplicación cuyo código está almacenado en un repositorio remoto. Los desarrolladores suben cambios al código todos los días, varias veces al día. Por cada envío al repo se ejecutan un conjunto de pasos automáticos que terminan construyendo una aplicación.

Estos pasos no solo arman la aplicación, ¡también la testean! Entonces...

La integración continua (CI) es una práctica de desarrollo que requiere que los desarrolladores integren código en un repositorio compartido varias veces al día. Al integrar contenido con cierta frecuencia, detectamos errores rápidamente y los localizamos de manera sencilla.

¿Qué son los triggers?

Un trigger es un “disparador”: algo que hace que una primera cosa active una segunda y así sucesivamente. Los triggers son parte de la familia de la “integración continua” o “CI” y se utilizan para ejecutar un pipeline mediante una llamada de una API u otro proceso en línea. Son sumamente útiles para concatenar procesos que, en líneas generales, constituirán un flujo de trabajo que servirá para construir algo.



¿Qué es un artefacto?

Varias veces hemos tenido que compilar código fuente para obtener un producto utilizable. Esos archivos resultantes del proceso de compilación se llaman artefactos.

Los artefactos son el resultado de la compilación la transformación de un código fuente en un archivo binario no modificable. Tenemos como ejemplo de ello archivos binarios: dll, jar, war, ear, msi, archivos exe, etc.

Mediante el uso de pipelines, puedo integrar el software administrador de artefactos para que almacene de forma centralizada mis artefactos y luego poder ser utilizados mediante llamadas en tiempo real.

CI/CD:

CI toma el producto final y lo hace disponible a la fase de testing. CD continúa el proceso y lo hace disponible a la fase de release.

La CI/CD incorpora la automatización continua y el control permanente en todo el ciclo de vida de las aplicaciones, desde las etapas de integración y prueba hasta las de distribución e implementación. Este conjunto de prácticas es una solución para los problemas que puede generar la integración del código nuevo a los equipos de desarrollo y de operaciones.

CD:

Continuous Delivery VS Continuous Deployment

El Despliegue Continuo (CD) puede referirse a dos prácticas similares pero con impactos diversos:

- Continuous Delivery.

- Continuous Deployment.

¡Una diferencia clave!

Los procesos de continuous delivery (o entrega continua) y continuous deployment (despliegue continuo) siguen las mismas etapas, ¡pero con una pequeña gran diferencia!

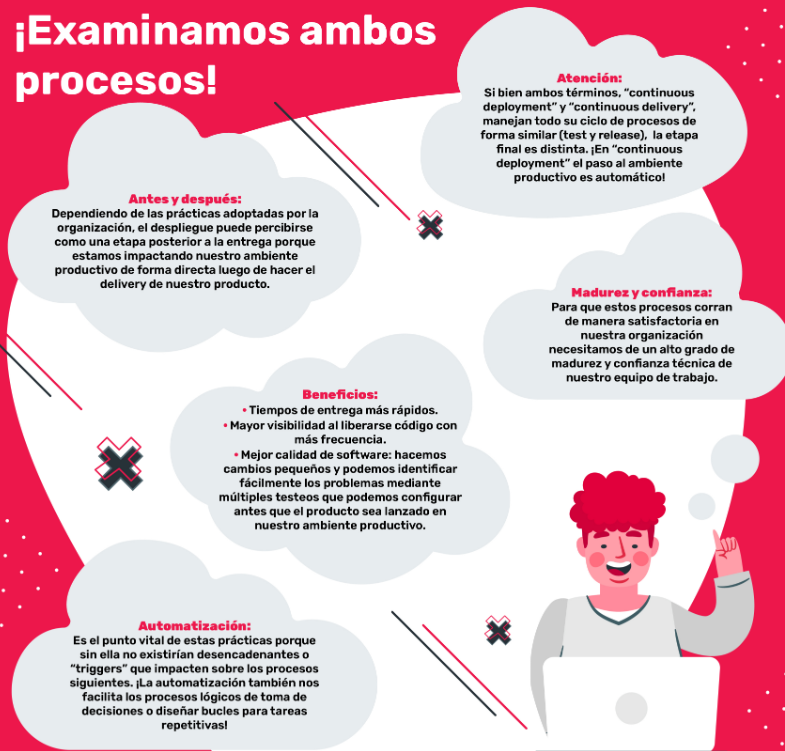
Continuous Delivery se focaliza en la automatización de los pasos para que nuestro software esté disponible para ser aplicado en los ambientes productivos en cualquier momento, ¡pero no hace la implementación automática en producción!

Continuous Deployment va un paso más allá. En esta práctica, todo es automático en los ambientes y también en producción. Buscamos que no exista intervención humana en ninguno de los procesos de trabajo.

Esta es la diferencia más importante con respecto a la entrega continua.

Para lograrlo, el pipeline de producción tiene una serie de pasos que deben ejecutarse en orden y forma satisfactoria ¡y de manera automática! Si alguno de esos pasos no finaliza de forma esperada, el proceso de despliegue no se llevará a cabo.





Pasos para un correcto despliegue de pipeline



Buenas practicas devOps para un pipeline ci/cd exitoso:

