DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Infraestructura II

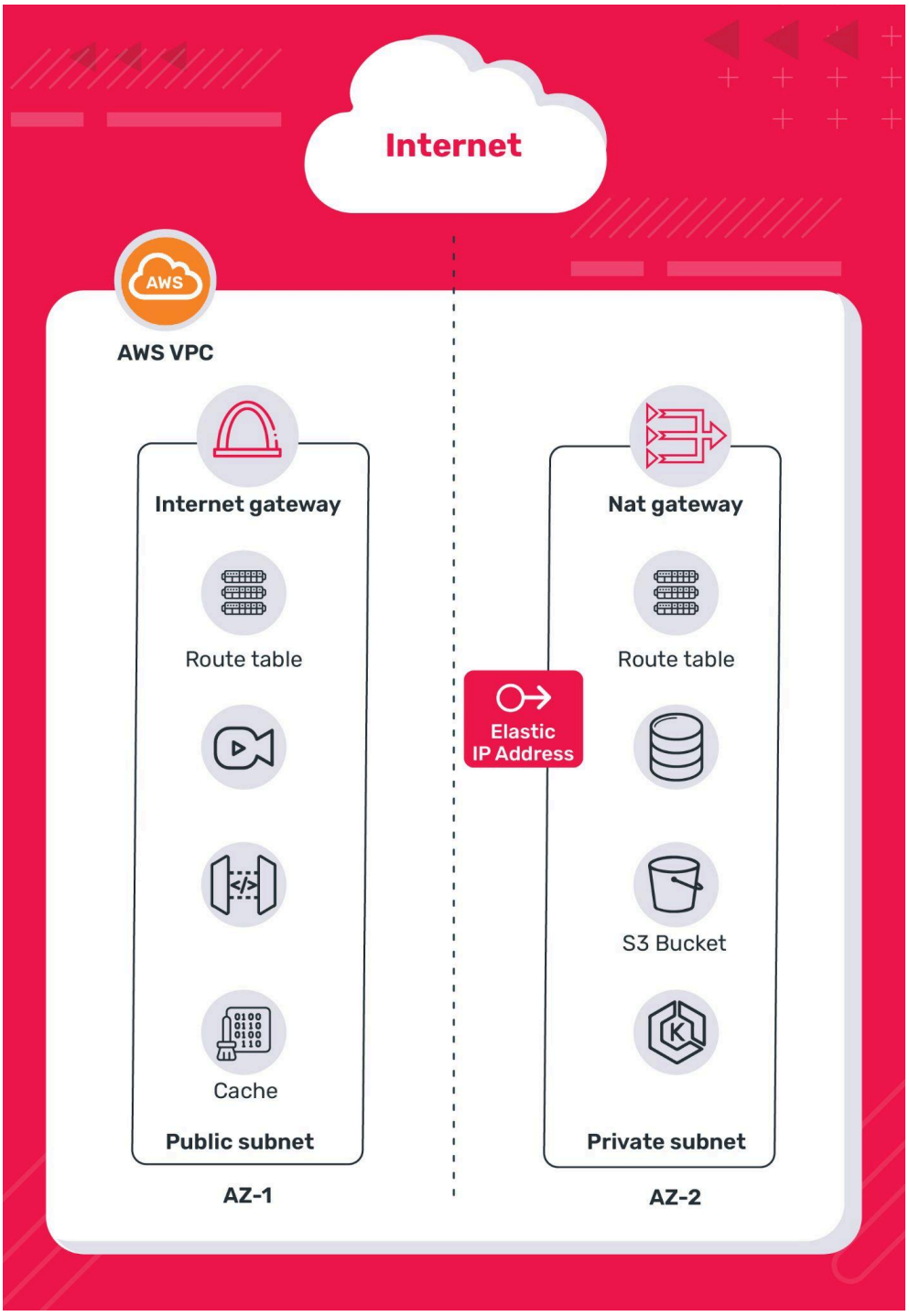
Administración de infraestructura con IAC y pipelines

Vamos a retomar una imagen que ya presentamos en la materia para revisar el trabajo que realizamos en Terraform. ¡Y nos movemos de herramienta! ¡De Jenkins a GitLab! En esta clase vamos a utilizar GitLab para levantar cada uno de los componentes que vemos en la imagen.

¡Vamos a continuar la práctica de la Clase 25 con GitLab CI para almacenar código, colaborar, *trackear* cambios, *versionar* y *deployar* usando un pipeline!

La infraestructura

¿Les resuena esta imagen? Este es un diagrama de arquitectura que les resultará conocido de la Clase 10 "Terraform continuado", específicamente de la práctica: "¡Levantamos una infraestructura de la vida real!".



En esa ejercitación, levantamos la estructura base donde luego se montó la capa de aplicación. Acá, hay una gran diferencia: En esta clase vamos a hacer lo mismo, pero

utilizando GitLab con el objetivo de mostrar como podemos integrar Terraform dentro de Continuous Integration.

Estos son los recursos de AWS que vamos a crear:

- 1 VPC
- 1 subnet pública y 1 privada
- 1 tabla de ruteo pública y 1 privada
- 1 asociación para la tabla de ruteo pública y 1 para la privada
- 1 elastic IP address
- 1 NAT gateway

¿Por qué lo hacemos de esta manera?

Pensemos en un ejemplo de la vida real. Vamos a contar un caso que se suele ver muy frecuentemente en prácticas empresariales. Como venimos trabajando, un pipeline no está netamente orientado al software de aplicaciones, ¡nos permite gestionar todo lo que pueda ser transformado en código!

Una práctica normal que suele verse en los perfiles SRE ("Site Reliability Engineer") o en los cloud engineers es la de administrar infraestructura en la nube. Ahora bien, como los recursos hardware en la nube son administrados solamente por personal propio de cada compañía, la única opción disponible es **utilizar Infraestructura como código**. Con

IAC, tendremos código de infraestructura para ser gestionado. **Esta gestión puede ser realizada mediante el uso de pipelines.**

¿Cómo sería esto?

En la vida real, llegará un punto en el que ya no tendremos necesidad de continuar levantando más ambientes. Es en este momento que nos podremos preguntar: "¿entonces ya no necesito GitLab y Terraform?". ¡SI y no! ¿Por qué? La respuesta se resolvería con un par de preguntas sencillas:

- ¿Cómo continuamos administrando la infraestructura?
- Si quiero crear nuevos usuarios, ¿cómo lo hago?
- Si necesito agregar o quitar instancias EC2, ¿qué utilizo?
- Si necesito abrir puertos TCP/UDP, ¿qué hago?

La respuesta puede ser muy sencilla: mediante la consola de administración web que nos ofrece AWS. ¿Verdad?

Bien, el caso es que esta opción es válida siempre y cuando nuestra infraestructura no sea muy grande, no necesite de cambios veloces, no haya necesidad de tener un seguimiento de los cambios realizados, no importen los errores humanos o no requiera de un control exhaustivo de sus recursos. Esto puede parecer razonable, sin embargo, en medianas y grandes empresas, y solo para citar un ejemplo; es casi imposible pensar de movida en un cambio lento, que demande horas para ejecutarse. **Pero entonces, ¿qué necesitamos?**

¡Automatizar!

La automatización es quizás el punto neurálgico donde confluyen escaneos de código, testeos, deployments, entre otros.

Administrar infraestructura como código (IAC) es fundamental.

Pero claro, esto ya lo vimos con Terraform. ¿Qué sería lo nuevo entonces? ¡Buena pregunta! Si lo que nos interesa es poder hacer cambios mediante la automatización: **¡Un pipeline dedicado a infraestructura es clave!**

En esta ejercitación estaremos levantando infraestructura mediante Terraform y GitLab. Pero más allá de la práctica, les hacemos la invitación a continuar investigando como "trabajar cambios" utilizando Terraform dentro de GitLab. ¡**Es clave seguir trabajando en cómo gestionar nuestra infraestructura una vez que está construida!**