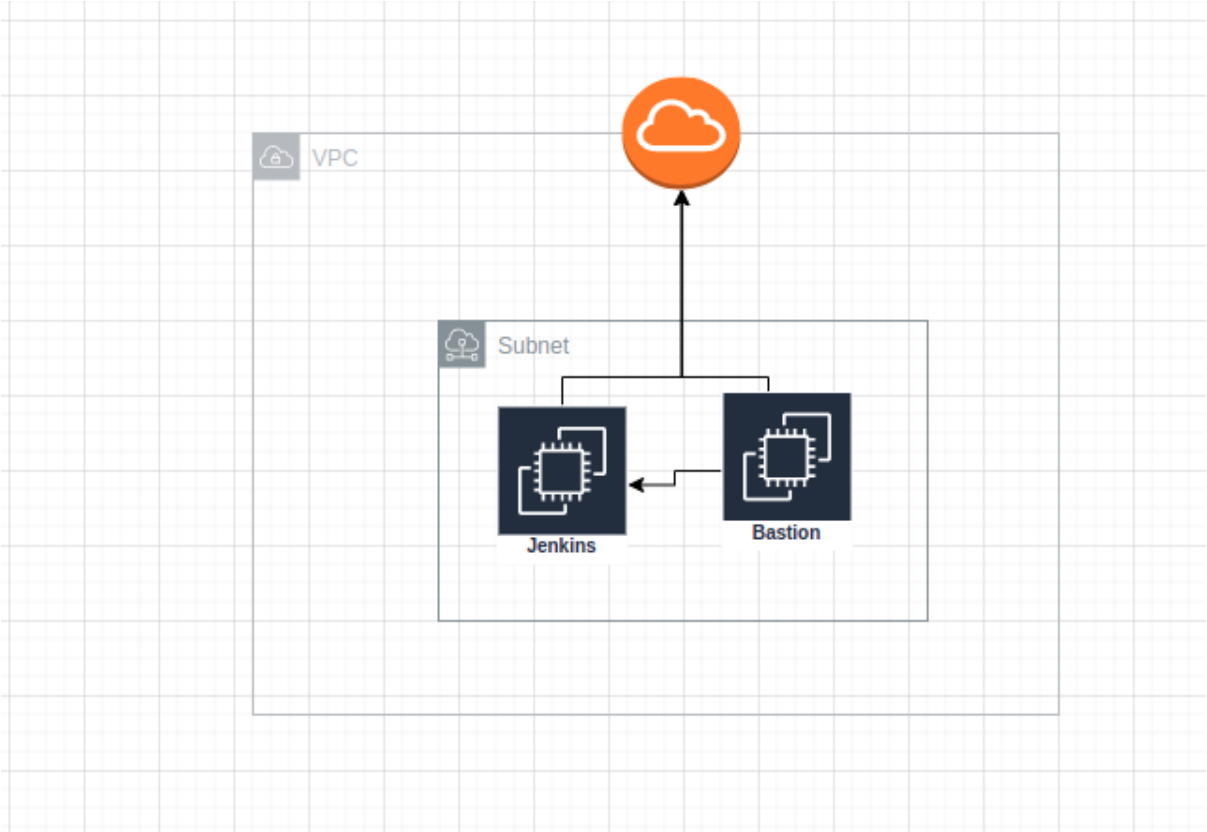
https://github.com/DiegoDutka/final_ac_-grupo2.git
Consulta a la API : /nro?nro1=7&nro2=3&ope=sum

INFRAESTRUCTURA

AWS



Para realizar la consigna decidimos levantar lo que es la parte de la infraestructura con Terraform.
Para esto levantamos dos máquinas virtuales

- Una que aloje el Jenkins
- Otra que sea Bastión para el manejo de seguridad

Ambas máquinas se encuentran en la misma VPC y subnet pública. Las dos cuentan con una internet gateway para que que poder acceder a internet
Para la parte de seguridad

- la máquina bastión se puede acceder solo por el puerto 22
- para la máquina Jenkins solo puede acceder por el puerto 22 el bastion y se puede acceder desde cualquier dirección por el puerto 80. Esta distinción de seguridad es debido a que cualquier persona tendría que poder ingresar a la pipeline

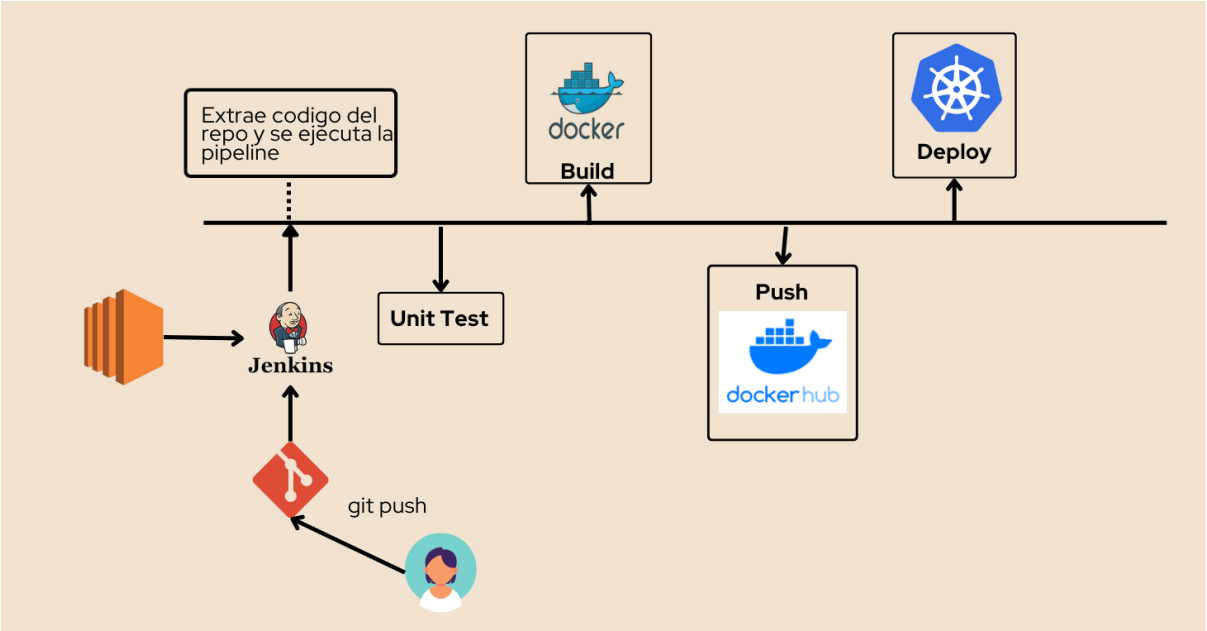
La configuración de Terraform consta de los siguientes archivos:

- varible.tf → aca se guardan las variables que son uso repetitivo
- networking.tf → en este archivo se relalizo la configuracion de:
    - la VPC
    - subnet pública
    - internet gateway
    - route table de subnet pública
- security.tf → configuracion de los sg de cada maquina
- instance.tf → configuracion de ambas instancias

La instalación de Jenkins y docker se hizo a través de un archivo yml que lo ejecuta el propio terraform al crear toda la infraestructura. La instalación de git, de kubectl y de aws cli se realizó de forma manual , pero se podría automatizar para unir todas las configuraciones.

PIPELINE

https://www.canva.com/design/DAFEXQ6cPpA/LHawuTIunYX7PVfhcAeLVg/edit?utm_content=DAFEXQ6cPpA&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton



Para la pipeline configuramos de la siguiente forma

1. el usuario hace un cambio al repo
2. Jenkins detecta este cambio entonces ejecuta la pipeline
3. Instala golang
4. Instala dependencias del programa
5. ejecuta unit test
    a. verifica una suma
    b. verifica si funciona una solicitud get
6. ejecuta la build construyendo la imagen de docker
7. pushea la imagen a docker hub
8. levanta esa imagen en un pod dentro de un cluster en eks

Realizamos una pipeline por cada stage, es decir tenemos una pipeline para QA, otra para producción y otra para staging. Donde cada una cuenta con su rama en el repo, por ende Jenkins detecta el cambio dentro de cada repo y ejecuta la pipeline Cabe destacar, que dentro de cada branch se encuentra la App, la configuración de deploy y el Jenkinsfile correspondiente.

S	W	Name ↓	Last Success	Last Failure	Last Duration	
✓	☁	pipeline_qa	1 hr 37 min #10	1 hr 40 min #9	1 min 14 sec	▷
✓	☁	staging	1 hr 36 min #18	1 hr 40 min #17	37 sec	▷
✓	☁	test	18 hr #19	19 hr #18	8.8 sec	▷
✓	☁	test_dockerfile	40 min #72	1 hr 57 min #68	1 min 26 sec	▷

**** test_dockerfile es la pipeline de produccion
** test no pertenece a ninguna stage

DEPLOY

El deploy de cada app, está creado con 2 pods idénticos y un load balancer de servicio. Previamente se realizó la creación del cluster mediante EKSCTL, creando 2 nodos.

En el tro repo tendria que estan el jenkinsfile, el yml y el dockerfile