Implementación desde docker-compose hacia kubernetes



Gabriela vargas

Electiva profesional 3

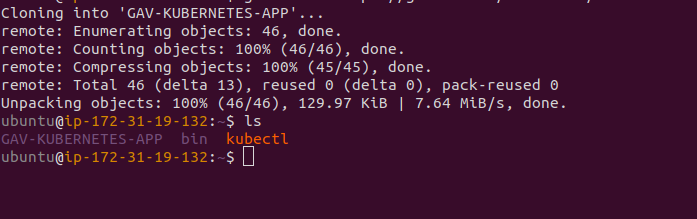
Universidad Libre Seccional Cali

2021

mi app de faker se encuentra formada por 3 servicios el servicio api , faker , y el servicio web, faker se conecta a una base de datos externa desplegada en heroku

para clonar el repositorio que contiene todo lo necesario ejecutar:

git clone https://github.com/gvargasv93/Kubernetes-App



antes de continuar instalaremos kubectl, minikube para crear nuestro cluster de kubernetes , instalaremos socat la cual es una herramienta de ubuntu necesaria para la comunicación entre los distintos pods que tendrá nuestro cluster , y partimos de que ya tenemos instalador docker en nuestra máquina ya que ya lo habíamos instalado en clases anteriores

**INSTALAR KUBECTL**

**curl -o kubectl** [**https://amazon-eks.s3-us-west-2.amazonaws.com/1.21.2/2021-07-05/bin/linux/amd64/kubectl**](https://amazon-eks.s3-us-west-2.amazonaws.com/1.21.2/2021-07-05/bin/linux/amd64/kubectl)

**sudo chmod +x ./kubectl**

**mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin**

**echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc**

**kubectl version --short --client**

con este comando verificamos que la instalación de kubectl se llevó a cabo correctamente

Ya teniendo instalado kubectl en nuestra máquina debemos instalar minikube

**Minikube** es una herramienta que administra máquinas virtuales en donde corre un cluster o mejor dicho una instancia de Kubernetes en un solo nodo. Sin embargo **Minikube** se apoya de un hypervisor

**curl -Lo minikube** [**https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64**](https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64) **&& sudo chmod +x minikube && sudo mv minikube /usr/local/bin/**

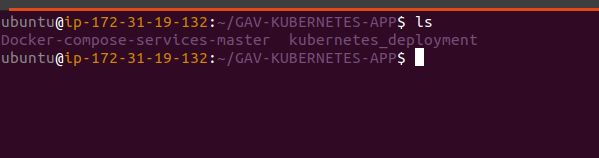
**minikube version**

**INSTALAMOS SOCAT**

**sudo apt-get install socat**

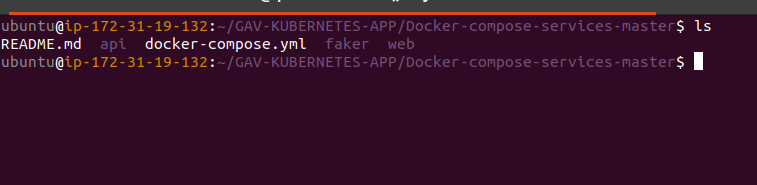
una vez clonado el repositorio de git debemos entrar a la carpeta GAV-KUBERNETES-APP

aquí tendremos 2 directorios el primero Docker-compose-services-master contiene todo el codigo de mi aplicacion realizada en docker-compose donde se comunican los 3 servicios y en kubernetes\_deployment se encuentran los manifiestos de kubernetes listos para desplegar los servicios en el cluster de minikube



inicialmente para realizar esta migración en mi app echa en docker compose no tenia subidas las imagenes a docker hub por lo cual tuve que hacer un build de cada imagen y subirlas a docker hub

para esto ejecute los siguientes comandos



como se puede observar en nuestra carpeta de docker-compose hay 3 carpetas una para cada servicio dentro de cada carpeta hice la construccion de la imagen y la subi a docker hub para esto entra a la carpeta api

**docker build -t api\_service .**

**docker tag api\_service:latest gvargasv/api\_service:latest**

**docker push gvargasv/api\_service:latest**

luego en la carpeta faker

**docker build -t faker\_service .**

**docker tag faker\_service:latest gvargasv/faker\_service:latest**

**docker push gvargasv/faker\_service:latest**

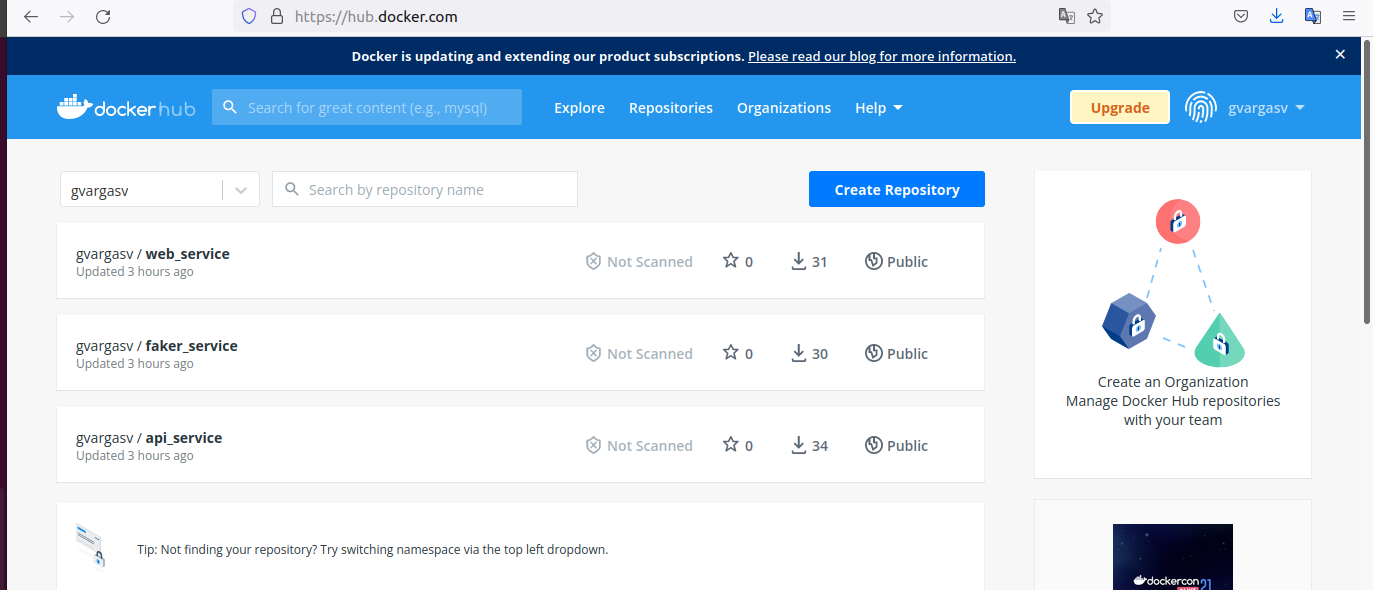
luego en la carpeta web

**docker build -t web\_service .**

**docker tag web\_service:latest gvargasv/web\_service:latest**

**docker push gvargasv/web\_service:latest**

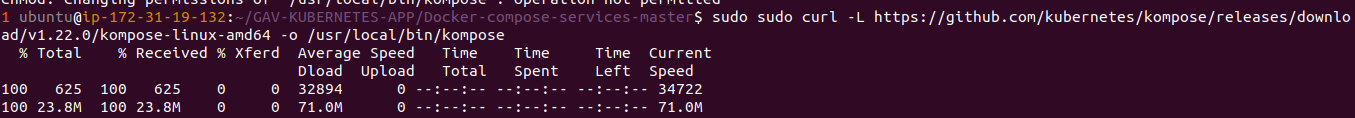
Una vez realizado esto ya tenia mis imagenes de cada servicio subidas en mi repositorio el cual se puede visualizar en el siguiente url



<https://hub.docker.com/u/gvargasv>

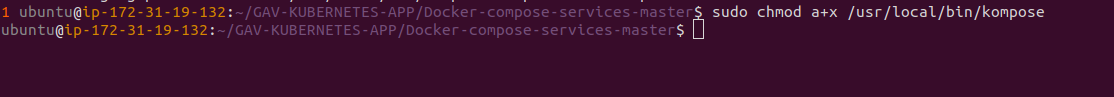
luego de esto ya podremos convertir nuestro archivo de docker-compose a los archivos de kubernetes para esto debo instalar la herramientas kompose

**sudo curl -L https://github.com/kubernetes/kompose/releases/download/v1.22.0/kompose-linux-amd64 -o /usr/local/bin/kompose**



y luego ejecutamos...

**sudo chmod a+x /usr/local/bin/kompose**



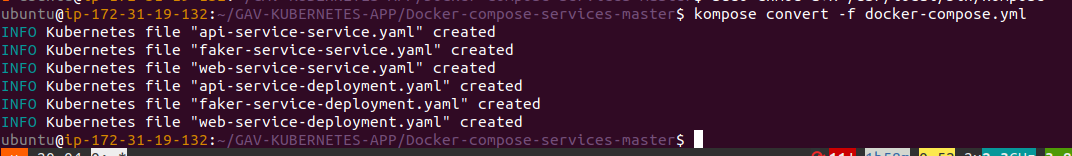
luego de esto debemos trasladarnos a la raíz de la carpeta en donde tenemos nuestro archivo docker-compose.yml y editarlo para que construya las imágenes a partir de nuestras imágenes en dockerhub y no haga el build a partir de las carpetas locales de nuestro proyecto



de esta forma nuestro docker-compose tomara las imagenes de docker-hub y no hara un build desde la carpeta local de nuestro proyecto para cada servicio

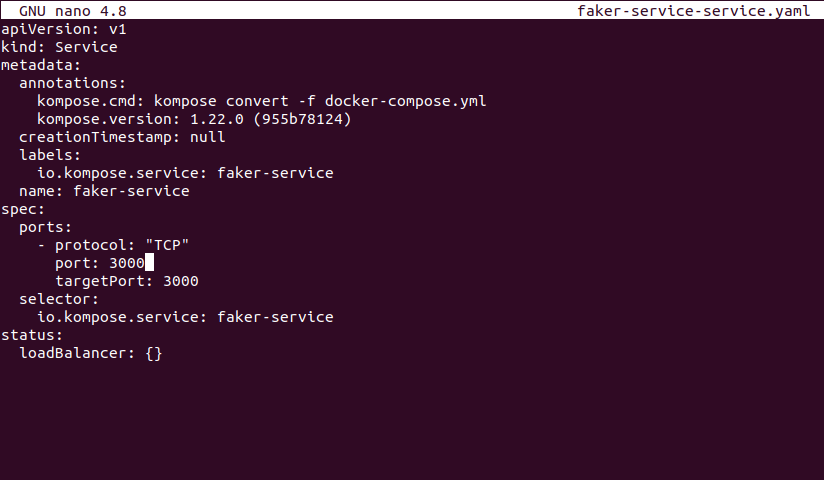
luego de esto estamos listos para migrar nuestra app de docker-compose hacia kubernetes para eso hacemos lo siguiente ejecutamos en la raíz donde se encuentra el archivo docker-compose.yml y ejecutamos el siguiente comando

**kompose convert -f docker-compose.yml**



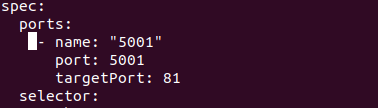
esto creará los archivos necesarios para hacer el deploy de nuestros servicios de kubernetes

ahora en cada archivo con el indicativo -service-service debemos editar lo siguiente

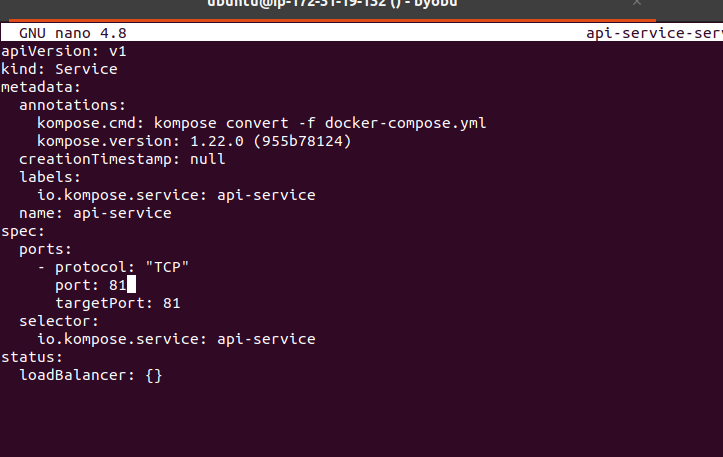


en la parte de ports: debemos especificar el protocolo y en port apuntamos el 3000 hacia el 3000 del contenedor

y lo mismo con los demás archivos ya que originalmente en este apartado despues de la conversión tendremos esto por ejemplo para el siguiente archivo el api-service-service.yaml



quedaria asi;



y luego para el archivo web-service-service debe quedar de esta forma



esto se hace con el fin de que apunte correctamente el puerto hacia nuestra app para cuando expongamos el servicio no tengamos problemas

entonces lo que acabamos de editar fueron estos 3 archivos:

**faker-service-service.yaml**

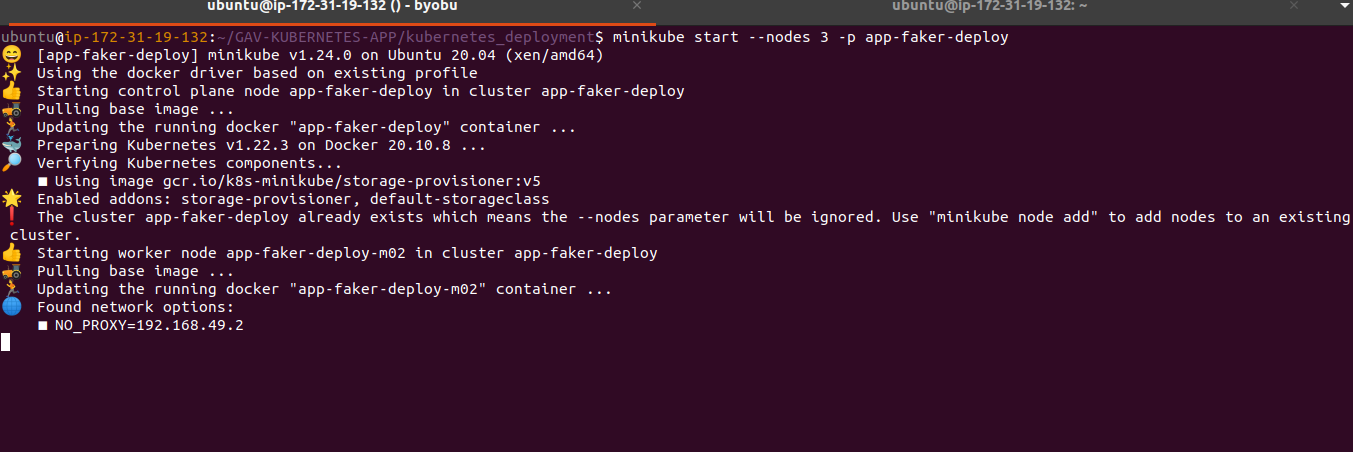
**api-service-service.yaml**

**web-service-service.yaml**

luego de esto ya con nuestros archivos .yaml en la raiz en donde estan nuestros archivos vamos a iniciar el cluster

para iniciar nuestro cluster de minikube ejecutamos lo siguiente:

minikube start --nodes 3 -p app-faker-deploy



Para subir nuestros servicios debemos aplicar los manifiestos de kubernetes finales de la siguiente manera

kubectl apply -f api-service-deployment.yaml

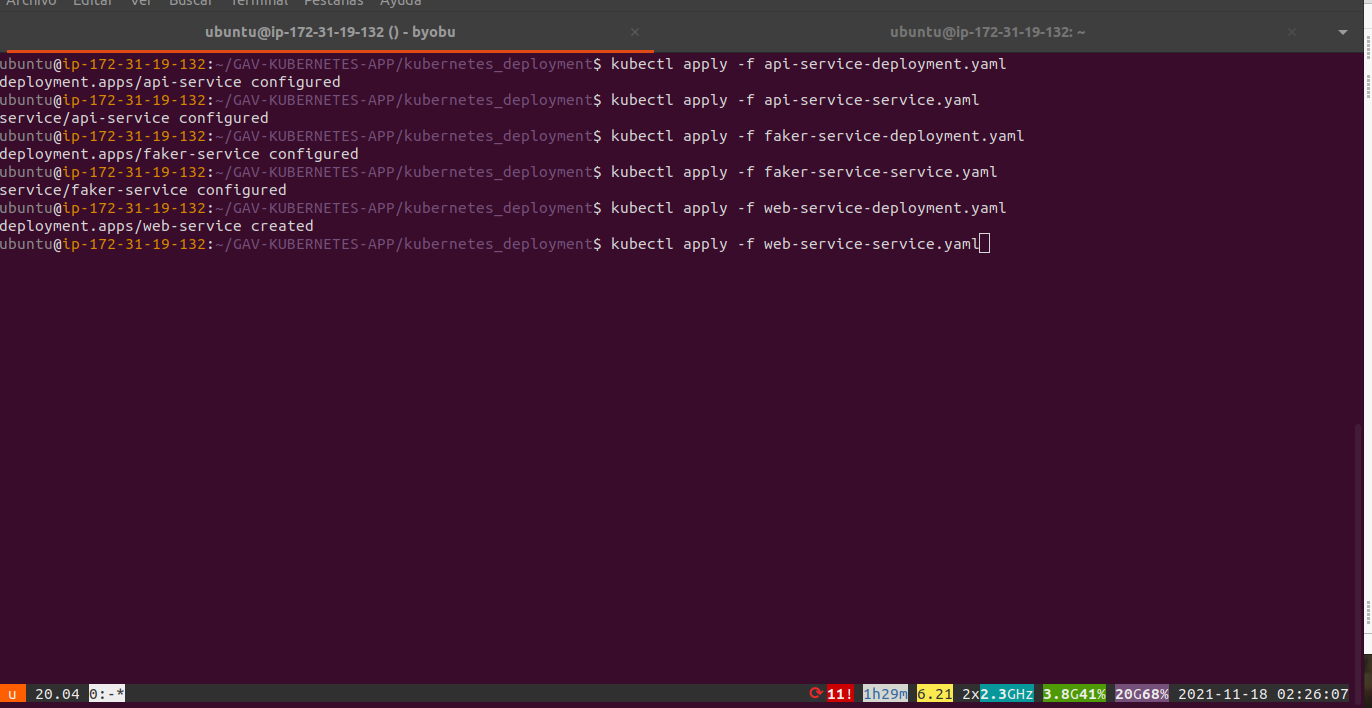
kubectl apply -f api-service-service.yaml

kubectl apply -f faker-service-deployment.yaml

kubectl apply -f faker-service-service.yaml

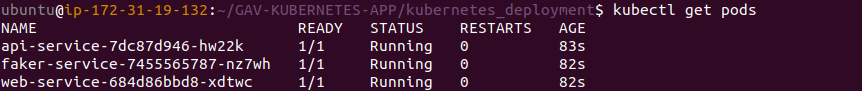
kubectl apply -f web-service-deployment.yaml

kubectl apply -f web-service-service.yaml



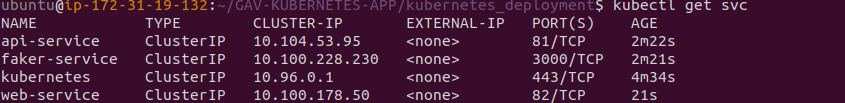
Luego debemos verificar que nuestros pods están corriendo normalmente con el comando

kubectl get pods



y también verificar que nuestros servicios están arriba

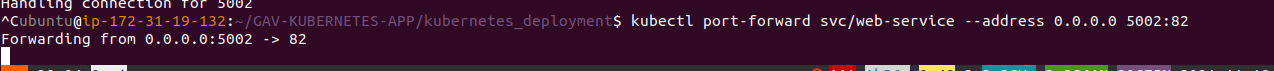
kubectl get svc



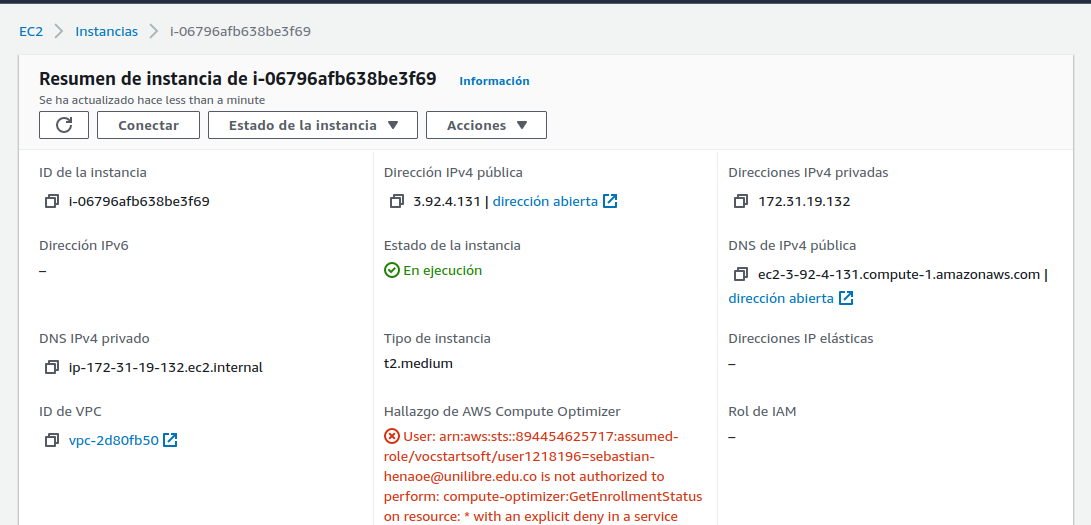
luego de esto debemos exponer nuestro web-service al exterior para poder acceder desde el navegador a nuestra aplicación usando la dirección ip pública de nuestra máquina esto lo hacemos con el siguiente comando

kubectl port-forward svc/web-service --address 0.0.0.0 5002:82

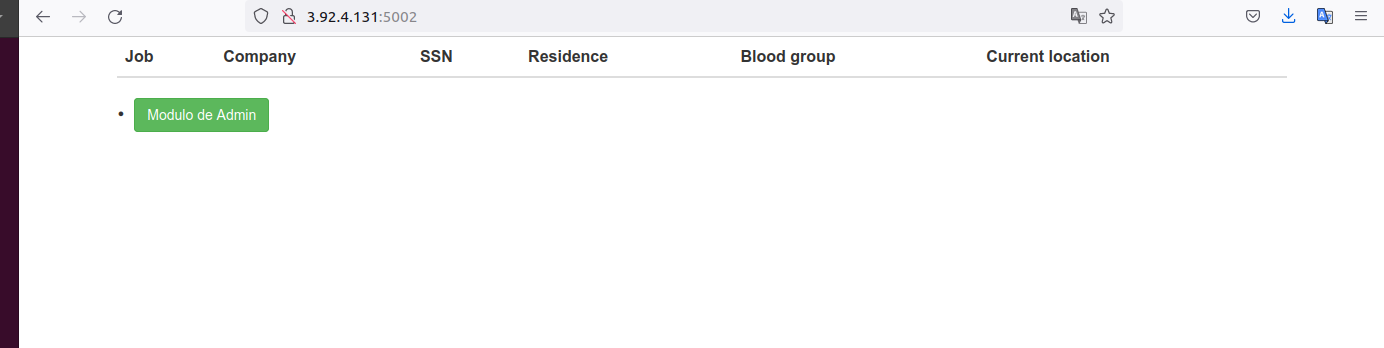
al hacerlo tendremos esta salida



y luego podremos ir a la ip pública en el puerto 5002 de nuestra máquina que se puede ver en el panel de instancias de amazon



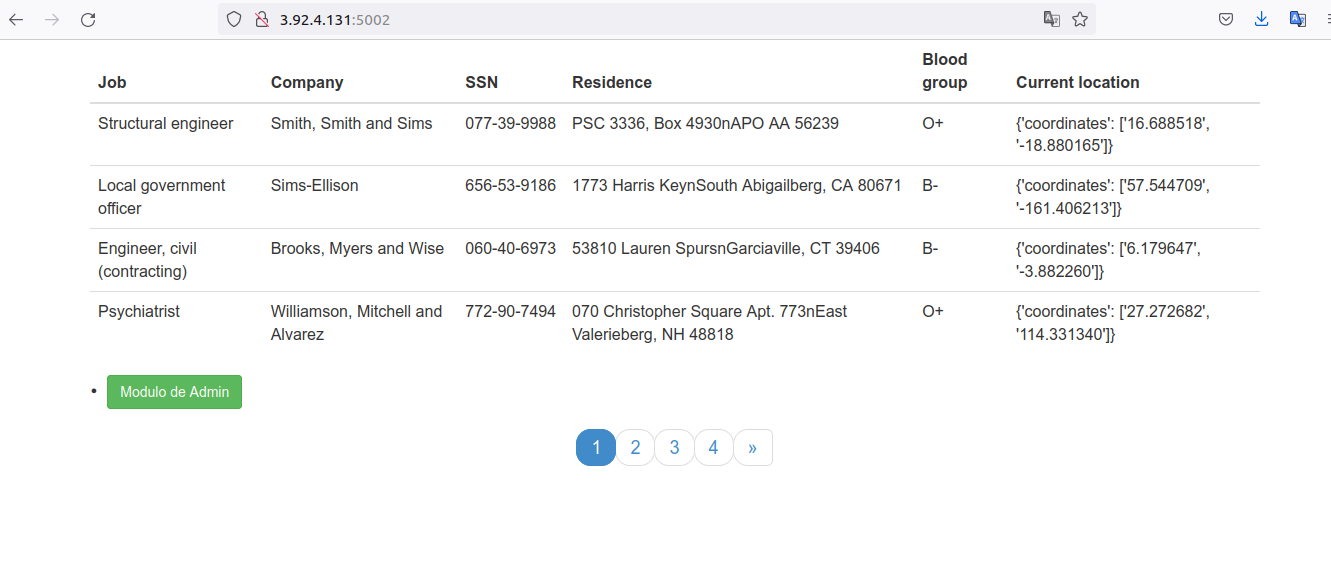
y podremos ver la app en ejecución



si damos click al módulo de admin



y damos click en generar datos fake y volvemos al Home



la app funciona correctamente desplegada en kubernetes utilizando 3 nodos