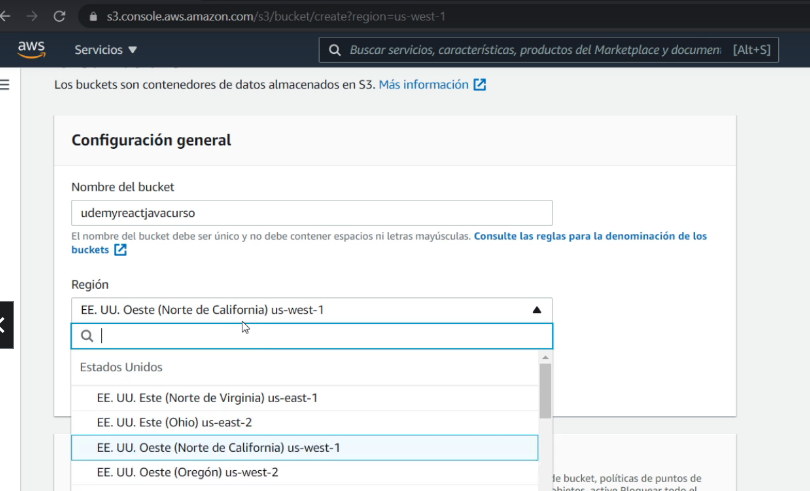
DESPLIEGUE DE LA APLICACIÓN

# Despliegue de la app React con AWS

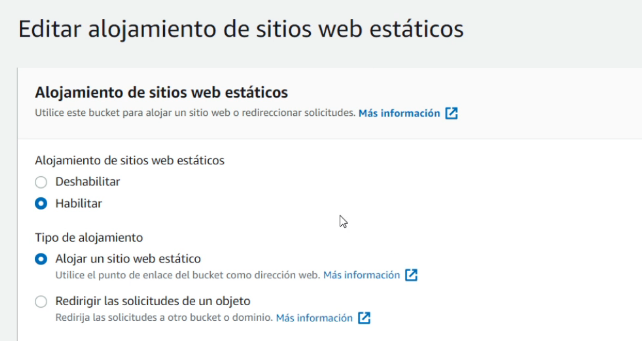
* Crear un bucket en AWS



Quitar el checkbox y control de versiones



* Configurar bucket
  + Ir a propiedades y en alojamiento de sitios web estáticos – habilitar





* Crear una política para el bucket

Ir al bucket y en permisos-Politica del bucket y editarlo colocando esto:

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "PublicReadGetObject",

"Effect": "Allow",

"Principal": "\*",

"Action": [

"s3:GetObject"

],

"Resource": [

"arn:aws:s3:::*<bucket-name>* /\*"

]

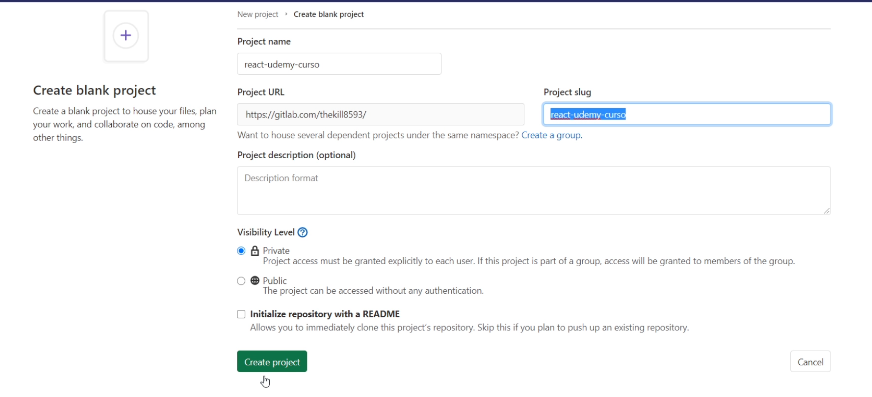
}

]

}

En bucket-name introducir el nombre del bucket que he creado

* Crear cuenta en GitLab y un repositorio



En la carpeta con la aplicación React:

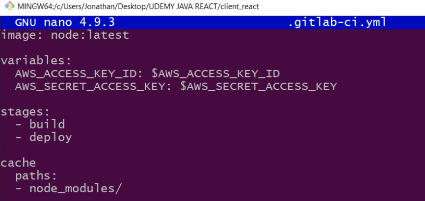
* Git remote add origin <url repositorio gitlab>
* Git add
* Git push origin master
* Archivo de configuración de despliegue continuo para gitlab

Crear archivo nano .gitlab-ci.yml

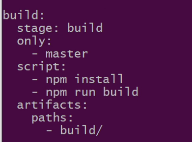


* + Primero especificar la imagen de Docker node js.
  + Segundo las variables de entorno para AWS.
  + Numerar las etapas:
    - Build y deploy
  + Definir que archivos cachear, por ejemplo el node\_modules para no volver a descargarlas
  + Definir el build:
    - stage y only para la rama que se desea construir.
    - Definir los scripts: npm install y npm run build para que se genere el archivo estatico de la app
    - Los artifacts que es donde se genera el archivo estático
  + Definir el deploy:
    - Stage y only para la rama que se desea desplegar
    - Definir la imagen de Docker: registry.gitlab.com/gitab-org/cloud-deploy/aws-base:latest
    - Definir las dependencias, carpeta build que es la carpeta que se genera al hacer el build
    - Definir el script: aws s3 cp –recursive ./build s3://<nombre\_bucket>

(lo que hace es copiar los archivos de build al bucket)

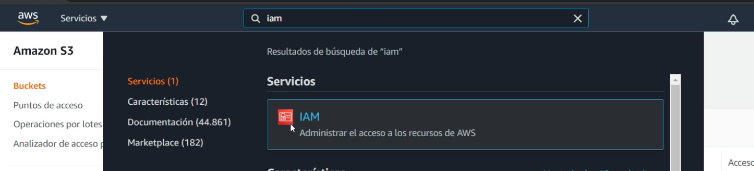


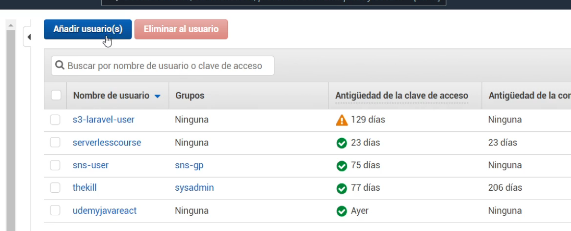
(cache con dos puntos)

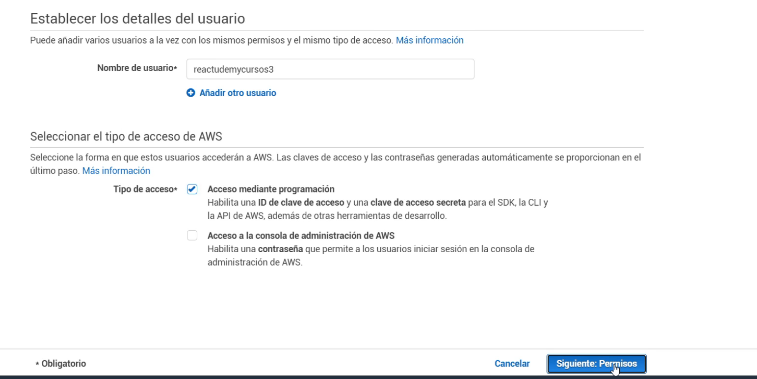
 

* Crear un usuario para S3

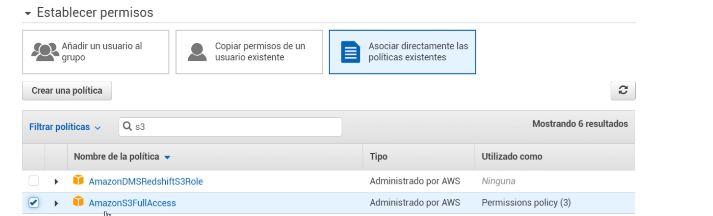
Panel IAM un servicio en AWS

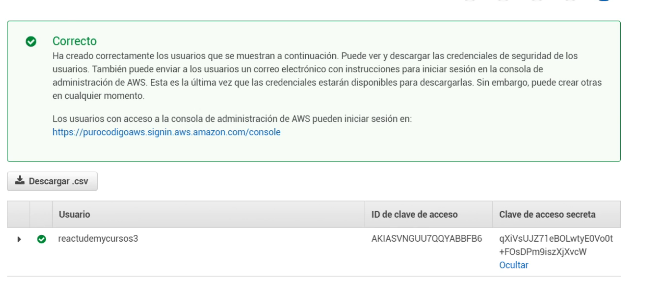






Seleccionar la política AmazonS3FullAccess

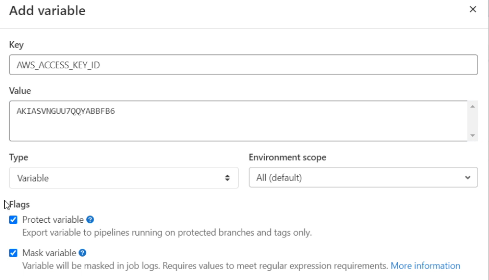


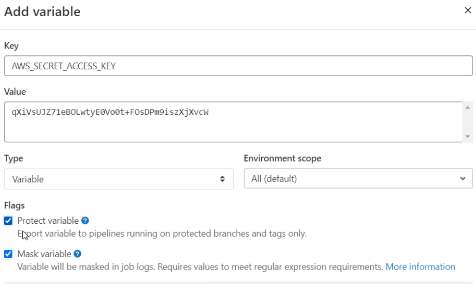


Ya está la clave de acceso y la clave secreta.

Ahora hay que crear las dos variables en gitlab:

* + Settings – CI/CD – Variables – Y se añaden estas dos variables:





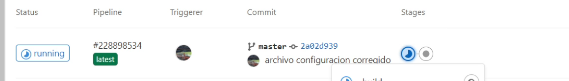
* Enviando cambios al repositorio remoto del archivo gitlab-ci.yml

Git add .

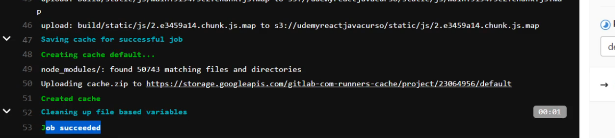
Git commit -m “<mensaje>”

Git push origin master



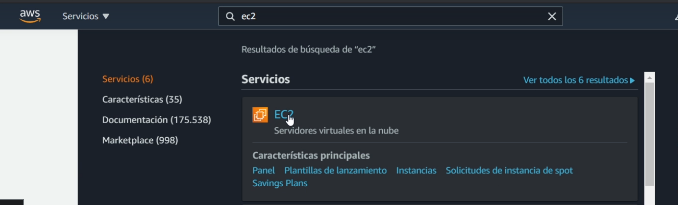


Esperar a que cargue todo:

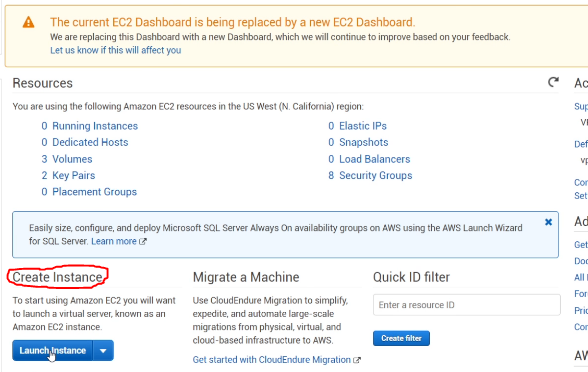


# Despliegue de la app Spring en AWS

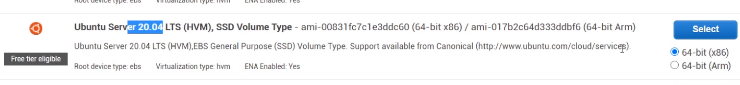
* Crear EC2 instance en AWS

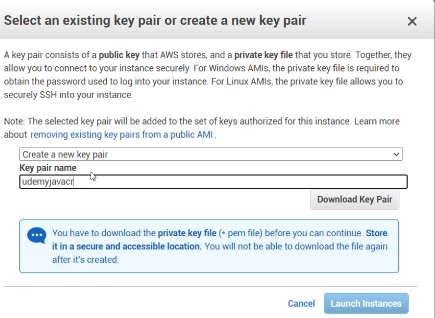


* + Lanzar nueva instancia

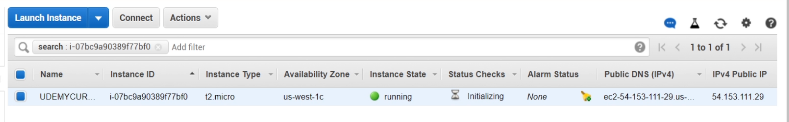


Seleccionar:





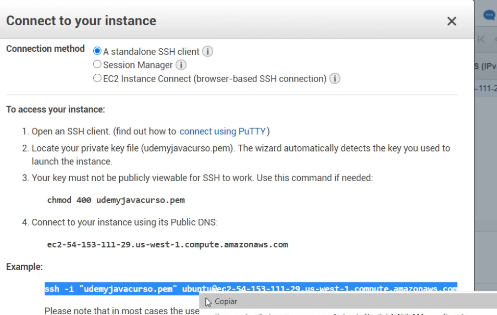
Descargar Key (IMPORTANTE) 🡪 Se descarga un archivo .pem para conectarse a la Máquina Virtual

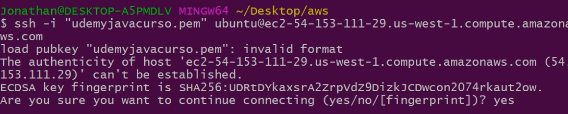


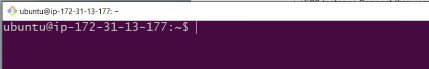
* Conectarse a la máquina virtual con ssh

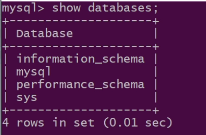
Tener el archivo .pem dentro de una carpeta llamada aws que creamos

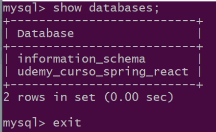
Copiar el comando ssh de la instancia







* Instalar MySQL
  + sudo apt-get update
  + sudo apt-get install mysql-server
  + sudo systemctl status mysql (verificar que este activo) sino ejecutar el comando: start mysql
  + sudo /usr/bin/mysql\_secure\_installation
  + sudo mysql -u root -p
  + 
* Crear usuario y base de datos
  + Sudo mysql -u root
  + créate database <nombre base de datos puesto en spring properties>
  + créate user ‘udemy’@’localhost’ identified by ‘udemy’;
  + Dar permisos: grant all privileges on <nombre base datos>.\* to ‘udemy’@’localhost’
  + Flush privilegies
  + Show databases



* + Cambiar el properties



* Crear repositorio Github

Git remote add “carpeta de la app spring”

Git add .

Git commit -m “first Commit”

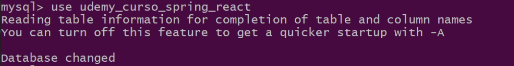
Git push origin master

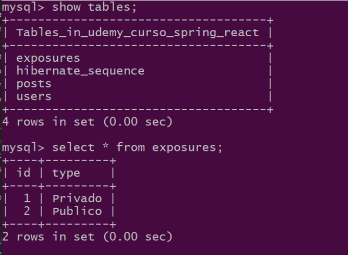
* Clonando repositorio en EC2
  + Mkdir java
  + Cd java
  + Git clone <url repositorio github>
  + Cd <carpeta clonada>
  + (Instalar Maven y java 11)

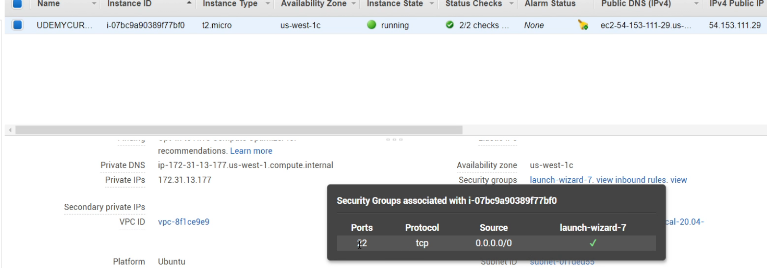
Sudo apt-get install openjdk-11-jdk

Sudo apt-get install maven

* Generar archivo para lanzar la aplicación
  + mvn install -DskipTests
  + cd target y aquí se encuentra el .jar
  + java – jar target/app.jar
  + mysql -u Udemy -p (se deberían crear ya las tablas en la base de datos)



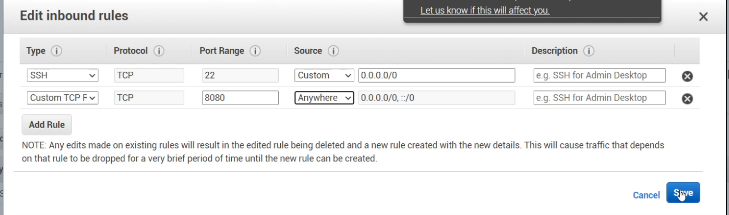




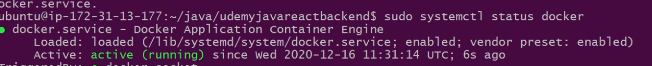
Ir a lauch-wizzard-7 para especificar el puerto que corre la aplicación



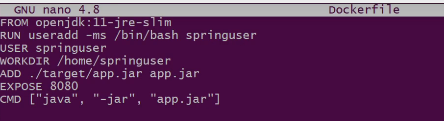
En inbound



* Instalar Docker y Docker-compose
  + Sudo apt-get install Docker.io
  + Sudo systemctl enable –now Docker
  + Sudo systemctl status docker

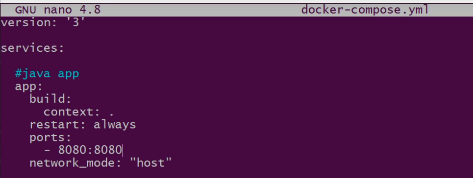


* + Sudo apt-install docker-compose
  + Sudo groupadd Docker
  + Sudo usermod -aG Docker $USER (para crear un usuario para Docker)
  + Crear el Dockerfile con nano Dockerfile



Traer la imagen openjdk de java, crear un usuario springuser y dar permisos en la consola bin bash, el directorio /home/springuser, añadir la app.jar en el puerto 8080 y ejecutar (CMD) el comando Java -jar app.jar

* + Crear docker-compose.yml



Context: .(significa el archivo DockerFile)

* Probar la API con Docker compose
  + docker-compose up
  + git push origin master (guardar cambios)