**CREANDO UN PIPELINE EN AWS PARA APP REACTIVA CONTENERIZADA CON DOCKER**

**ARQUITECTURA AWS:**

Diagram

Description automatically generated with medium confidence

PRE-REQUISITOS:

* Tener una cuenta de AWS
* Tener un usuario con los permisos necesarios en IAM a servicios como:
* CodeCommit
* CodeBuild
* CodeDeploy
* ECS
* EC2

1. CODE COMMIT SERVICE.

Crear un repositorio en CodeCommit

Buscamos el servicio CodeCommit y hacemos click en “Create repository”

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, website

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Una vez creado el repositorio es necesario configurar nuestra maquina local para poder hacer pull/push a un repositorio de CodeCommit para esto hacemos Click en SSH y según tu Sistema Operativo sigues los pasos (ejemplo para Windows)

Graphical user interface, text, application

Description automatically generated

Luego de Pushear el código deberás ver el código en repositorio remoto de CodeCommit.

Graphical user interface, text, application, email

Description automatically generated

1. CODE BUILD SERVICE

Crear un proyecto en CodeBuild

Busca el servicio de CodeBuild y luego click en “Crear Build Project”

A screenshot of a computer

Description automatically generated

Graphical user interface, application

Description automatically generated

Agrega un nombre y una descripción

Graphical user interface, text, application, email

Description automatically generated

Selecciona la fuente de la que se obtendrá el código fuente (Para nuestro caso, el repositorio de CodeCommit creado anteriormente)

Graphical user interface, text, application, email

Description automatically generated

En la sección de Environment, lo configuramos así:

Graphical user interface, text, application, email

Description automatically generated with medium confidence

El resto se debe dejar tal cual está y finalmente click en “Create build Project”

Sin embargo, para que el build funcione, es necesario crear un archivo buildspec.yml en la raíz de nuestro proyecto, así que lo creamos.

Graphical user interface, text, application

Description automatically generated

Y hacemos commit & push al repositorio de CodeCommit.

Posteriormente, podemos validar que el Build Project este bien haciendo click en “Start Build”, el build será exitoso como se muestra a continuación:

A screenshot of a computer

Description automatically generated

1. CODE PIPELINE SERVICE

Ahora pondremos todos los servicios automatizados creando un proyecto de “Code Pipeline” el cual se encargará de:

* 1. Leer los push que hagamos al repositorio de Code Commit
  2. Disparar un trigger que inicie un nuevo build en Code Build
  3. Crear un stack en CloudFormation
  4. Disparar un trigger que inicie un service de Amazon ECS
  5. Desplegar la solución en un Amazon EC2.

Buscamos el servicio Code Pipeline y hacemos click en “Create Pipeline”

Graphical user interface, text, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Escribo un nombre para el pipeline y click en Next

Graphical user interface, text, application, email

Description automatically generated

Seleccionamos la fuente de donde sacaremos el código fuente (Para nuestro caso nuestro proyecto y rama de Code Commit) y hacemos click en Next

Graphical user interface, text, application

Description automatically generated

En el siguiente paso seleccionamos el build Project creado con anterioridad y click en Next.

Graphical user interface, application

Description automatically generated

Por ahora le daremos click al botón de “Skip Deploy” (Este Stage lo configuraremos más adelante) pues solo queremos verificar que cuando hagamos un push a la rama de nuestro repositorio, se ejecute el pipeline y dispare el build.

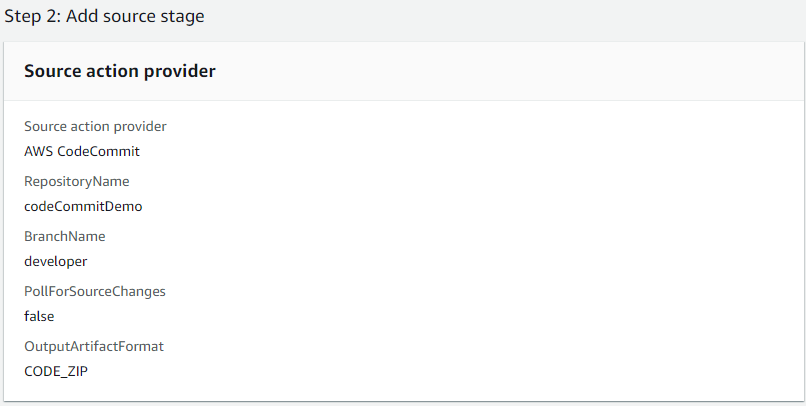
Graphical user interface, text, application, email

Description automatically generated

En el siguiente paso “Review” verificamos que todo esté bien y finalizamos haciendo click en “Create Pipeline”

Graphical user interface, text, application, email

Description automatically generated



Graphical user interface, application, Word

Description automatically generated

A picture containing diagram

Description automatically generated

Una vez creado, se ejecutará el pipeline exitosamente.

Graphical user interface, text, application, email

Description automatically generated

1. AMAZON ECR SERVICE

Ahora debemos crear un repositorio para almacenar nuestras imágenes de Docker en Amazon ECR (Amazon Elastic Container Registry), para eso busco el servicio ECR y click en crear repositorio (Get Started)

Graphical user interface, text, application

Description automatically generated

Seleccionamos privado/publico según sea necesario, le asignamos un nombre y click en “save”

Graphical user interface, text, application, email

Description automatically generated

Una vez creado nuestro reposiorio en ECR, hacemos click en “View Push Commands” y en este popUp pódras ver los comandos necesarios para logearte en ECR, hacer el build de la imagen, taggearla y pushearla a ECR

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Con los comandos anteriores debemos actualizar el buildspec.yml construido en el paso 2 de CodeBuild así:

Graphical user interface, text, application

Description automatically generated

Una vez esté listo el builspec.yml pusheamos el cambio al repositorio de CodeCommit para que el pipeline se ejecute automáticamente, el stage de Build deberá mostrar la siguiente excepción:

Graphical user interface

Description automatically generated

Y esto se debe a que nuestro IAM-User no tiene los privilegios necesarios, así que se los debemos asignar así:

Buscamos el servicio de IAM y en el menú de la izquierda click en “Roles”

Graphical user interface, text, application, email

Description automatically generated

En search buscamos el rol que nos muestra el mensaje de error en los logs del build para nuestro caso “codebuild-employee-api-build-service-role”

Graphical user interface, text, application, email

Description automatically generated

Click en el role y luego click en “Attach Policies” y buscamos la política: “AmazonEC2ContainerRegistryPowerUser”

Graphical user interface, application

Description automatically generated

La seleccionamos y click en Attach Policy

Ejecutamos de nuevo el stage de Build del Pipeline y este será exitoso

Graphical user interface, application, Word, Teams

Description automatically generated

1. AMAZON ECS SERVICE

En ECS (Elastic Container Service) crearemos 3 componentes:

* + Una “Task Definition” la cual tendrá la configuración de los contenedores que serán desplegados y como se comunican entre ellos.
  + Un Cluster que definirá en donde se desplegaran los contenedores de la “Task Definition”, para nuestro ejemplo EC2
  + Un “Service” el cual asocia el cluster con la task definition

5.1 ECS TASK DEFINITION.

Ahora debemos crear una nueva “Task Definition” en ECS.

Graphical user interface, text, application

Description automatically generated

Seleccionamos EC2 y click en Next

Graphical user interface, application

Description automatically generated

Definimos un nombre y un “network mode”

Graphical user interface, text, application

Description automatically generated

Principales “Task Settings” para el ejemplo:

* Dos contenedores: employee-api (Back end en Webflux) y MongoDB de las imágenes públicas de ECR
* Conectarse a mongo via network
* Mappear el puerto del host al puerto del contenedor para employee-api
* Settear la varable DB\_URI

**Primer Contenedor:**

Ingrese el nombre: employee-api y defina la URL de su repo de ECR

Graphical user interface, text, application, email

Description automatically generated

En la parte de “Environment Variables” y “Network Settings” debemos configurar el DB\_URI y el link al contenedor de Mongo (No definido aun)

Graphical user interface, text, application, email

Description automatically generated

Click en add

**Segundo Contenedor:**

Al igual que con el primer contenedor, defina el nombre y URL con los siguientes valores:

Nombre: mongo-employee-api-container

URL: public.ecr.aws/bitnami/mongodb:latest

Graphical user interface, text, application, email

Description automatically generated

Y en la parte de abajo en “Network Settings” definimos el HostName, como se estableció en la creación del repo del back, para nuestro ejemplo: mongo-employee-api-container

Graphical user interface, application

Description automatically generated

Click en Add.

Después de añadir dos contenedores a la tarea, cree la tarea y verás una pantalla similar a la que se muestra a continuación

Graphical user interface

Description automatically generated with medium confidence

* 1. ECS CLUSTER

Ahora debemos crear el cluster en ECS (Elastic Container Service) para eso buscamos el servicio ECS y click en “Clusters” en el menú de la izquierda y luego “Create Cluster”

Graphical user interface, application, Teams

Description automatically generated

Seleccionamos el que dice EC2 Linux + Networking y click en “Next step” (Internamente este Cluster creará por nosotros una VPC y unas Subnets)

Graphical user interface, application

Description automatically generated

Ahora es necesario configurar el Cluster, para esto definimos un nombre de cluster y establecemos el tipo de instancia EC2 donde se desplegará nuestra solución.

Graphical user interface, text, application, email

Description automatically generated

El resto lo dejamos por defecto y click en “Create”.

Una vez termine de crear el Cluster debemos poder verlo sin services o task definitions.

Graphical user interface, application

Description automatically generated

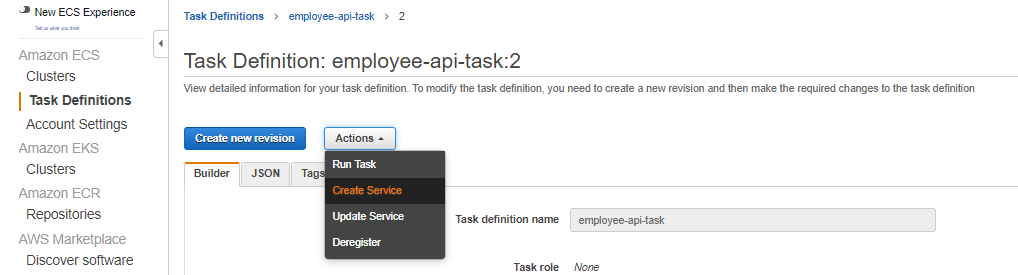
Sin embargo, la instancia de EC2 deben estar arriba, para verificar buscamos el servicio EC2 Y hacemos click en instances.

Graphical user interface, text, application

Description automatically generated

5.3 ECS SERVICE

El último paso es crear un servicio que tomará la tarea y el clúster y hará que los contenedores de la tarea se ejecuten en el clúster especificado. Para eso vamos al menú de la izquierda y en Task Definition -> employee-api -> Actions -> Create Service.



Y configuramos el servicio:

Graphical user interface, application

Description automatically generated

Click en siguiente y finalmente en “Create”

Graphical user interface, application

Description automatically generated

**TODO ESTA LISTO.**

Todo debería estar listo ya. Para verificarlo, necesitamos buscar la IP pública o el DNS público de la instancia de EC2. Para ello, haga clic en Clusters -> employee-api-cluster (nombre del cluster) -> Instancias ESC (pestaña) y Container instance:

Graphical user interface, application

Description automatically generated

Y ahora podremos hacer peticiones a nuestra aplicación a través de la IP Publica de la instancia del EC2

Graphical user interface, text, application, chat or text message

Description automatically generated

Y tendremos el resultado esperado:

Graphical user interface, text, application, website

Description automatically generated

Via Postman:

Graphical user interface, text, application, email

Description automatically generated

1. CREAR EL STAGE DE DEPLOY EN EL PIPELINE

Buscamos el pipeline creado anteriormente.

Graphical user interface, text, application, email

Description automatically generated

Click en el pipeline y luego en editar y abajo del stage de Build agregamos un nuevo stage con el nombre de Deploy

Graphical user interface, text, application

Description automatically generated

Y lo configuramos así:

Graphical user interface, text, application, email

Description automatically generated

Del ScreenShot anterior es **importante** el image definitions file, el cual es un archivo que genera el build y sirve para establecer el nombre de la tarea que debe ejecutar el Cluster->Service de ECS, así que es necesario modificar el buildspec.yml así:

Graphical user interface, text, application

Description automatically generated

Una vez pusheemos este cambio al repositorio de CodeCommit el pipeline se ejecutará en su totalidad exitosamente.

Teams

Description automatically generated with low confidence

Y los cambios que hagamos se verán reflejados en nuestra instancia de EC2

Graphical user interface, text, application, website

Description automatically generated

ESO ES TODO, HEMOS ACABADO DE CONFIGURAR NUESTRO PROYECTO EN AWS!!!

A close up of a sign

Description automatically generated with low confidence