Despliegue de la aplicación BlueSky en la plataforma de la nube AWS

Taller de Aplicaciones Sociales – UNMSM

Alumno:
- Quispe Richar

12/07/2024

Contenido

1. Objetivos del proyecto

- Implementación de una instancia de Bluesky en la nube de AWS.
- Aplicación de la tendencia Infraestructura Como Código para la automatización del despliegue.
- Aplicar pruebas de testing para evitar errores en la construcción de la aplicación.

2. Descripción general del proyecto

Este documento detalla los procedimientos necesarios para implementar la red social BlueSky en AWS, utilizando los servicios de IAM, EC2 y ECR, y automatizando el flujo de trabajo mediante GitHub Actions. Además, se presenta un método para automatizar este despliegue sin necesidad de interacción directa con la interfaz web de AWS.

La Figura 1 ilustra el flujo de trabajo para el despliegue continuo en AWS, donde GitHub Actions se utiliza para automatizar la implementación del código en ECR. Sin embargo, IAM y EC2 deben configurarse manualmente.

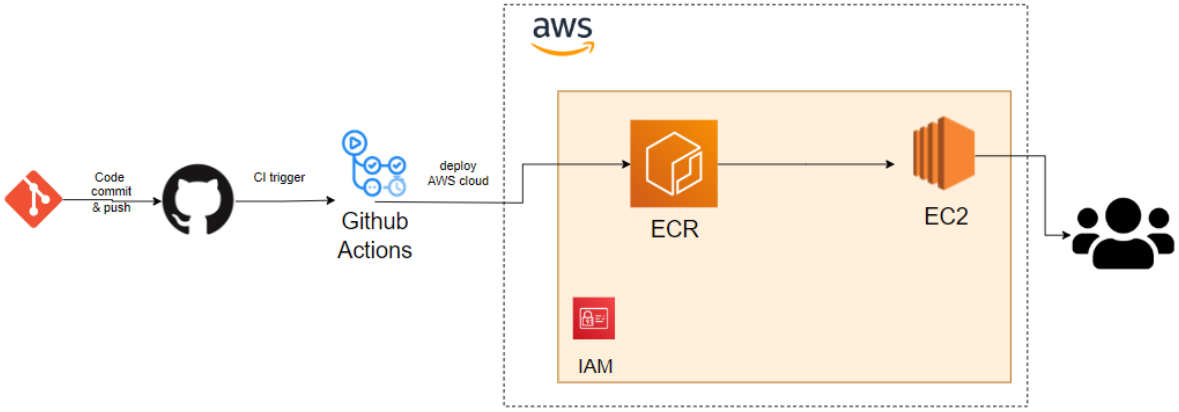

Figura 1: Arquitectura de despliegue

3. Testing con GitHub Actions

Para el proceso de pruebas existen dos flujos de trabajo, como se observa en la Figura 2.

3.1. Golang test

Este flujo de trabajo de GitHub Actions, definido en el archivo **golang-test-lint.yaml**, se activa con eventos de *pull request* y *push* a la rama `main`, y se ejecuta concurrentemente para realizar tareas de construcción, prueba y linting en un entorno Ubuntu. Consta de dos trabajos: `build-and-test` y `lint`.

En `build-and-test`, primero se realiza la comprobación del código mediante `make check`, luego se compila el binario con `make build` y, finalmente, se ejecutan las pruebas con `make test` después de configurar Go y simular la presencia de archivos estáticos.

En el trabajo `lint`, después de configurar Go y simular archivos estáticos, se ejecuta `make lint` para analizar el código en busca de problemas de estilo y calidad.

3.2. Lint

Este flujo de trabajo de GitHub Actions, llamado **lint.yml**, se ejecuta en cada *pull request* y *push* a la rama `main`. Tiene dos trabajos: `linting` y `testing`. En el trabajo `linting`, se revisa el código descargado del repositorio, se instalan dependencias con Yarn y se ejecutan varios checks: linters, Prettier, compilación de internacionalización y verificación de tipos. En el trabajo `testing`, también se descarga el código y se instalan dependencias, se compila, y luego se ejecutan pruebas con el entorno de Node ya configurado.

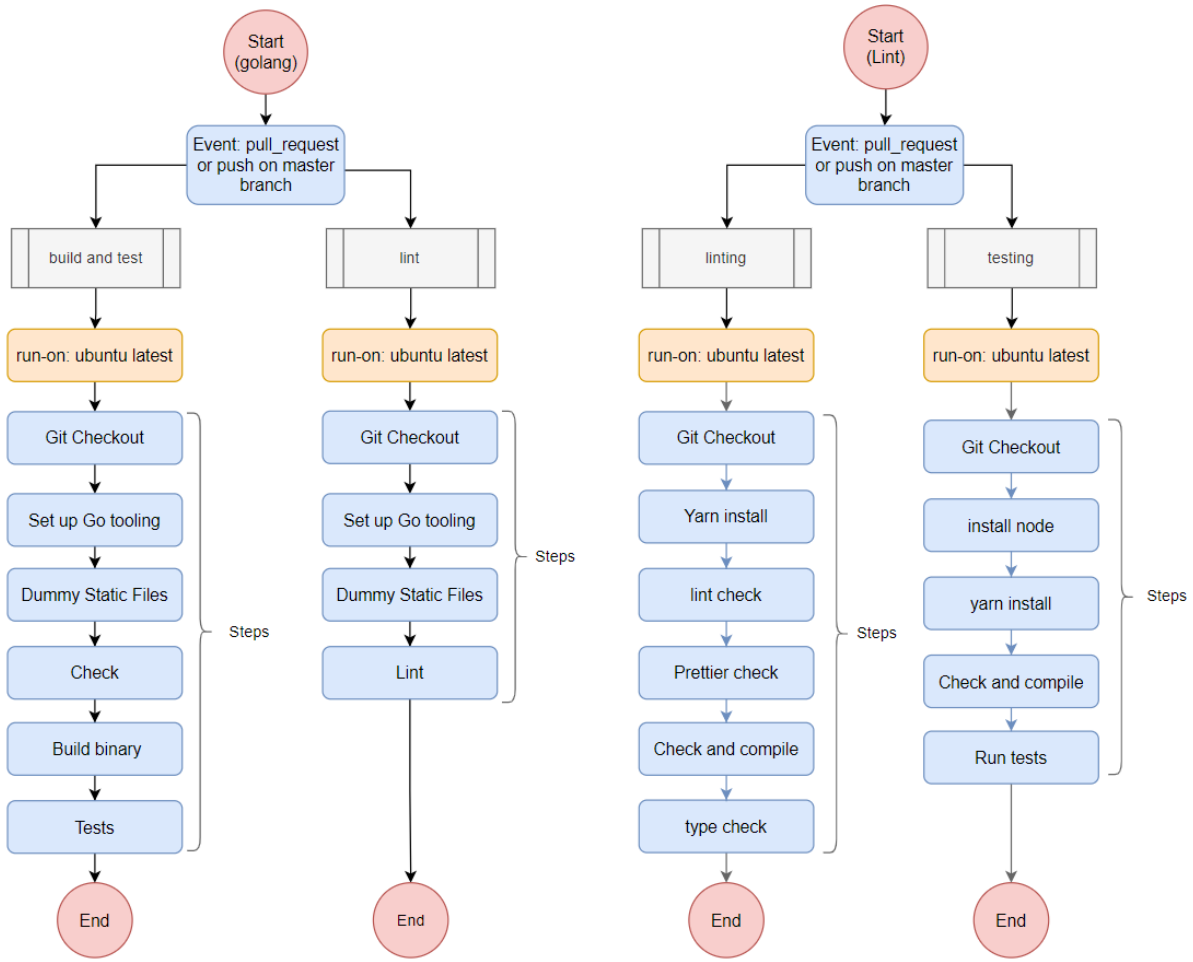

Figura 2: Workflows de testing

4. Configuración de AWS

4.1. Creación de una cuenta IAM

Para configurar el entorno en la nube, es imprescindible tener una cuenta en AWS. Dentro de AWS, el servicio IAM (Identity and Access Management) permite la creación de usuarios con los permisos necesarios. Resulta fundamental crear un usuario específico y asignarle los permisos adecuados. Además, se debe generar una clave de acceso para autenticar al usuario mediante **AWS CLI** y **Terraform**.

Como se muestra en la Figura 3, se ha concedido acceso total a los servicios de AWS con el propósito de evitar cualquier problema relacionado con autorizaciones.

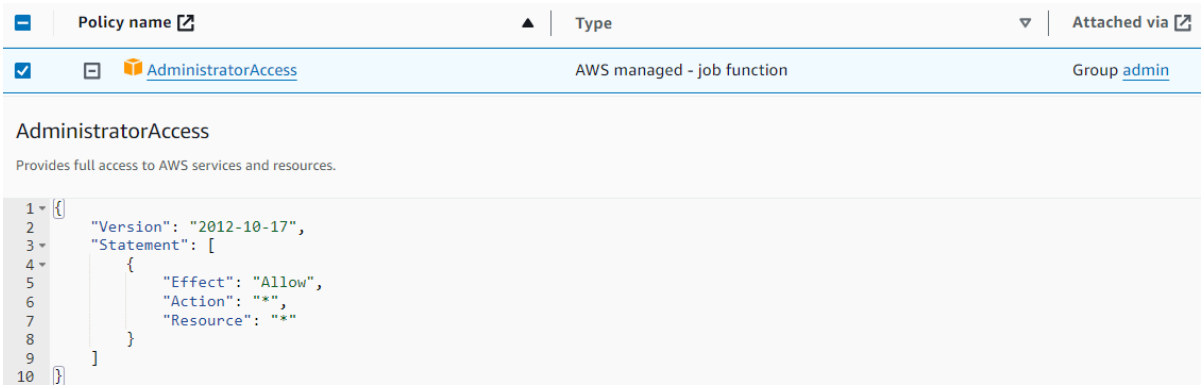

Figura 3: Politica de acceso completo en AWS

4.2. Configuración ECR

Amazon ECR (Elastic Container Registry) es un servicio de AWS para almacenar, gestionar y desplegar contenedores Docker.

En esta apartado, se creará un repositorio privado con el nombre bskyweb. Es crucial que el repositorio sea privado y tenga exactamente este nombre, ya que cualquier discrepancia provocará errores en el flujo de trabajo configurado en GitHub, como se ilustra en la Figura 4.

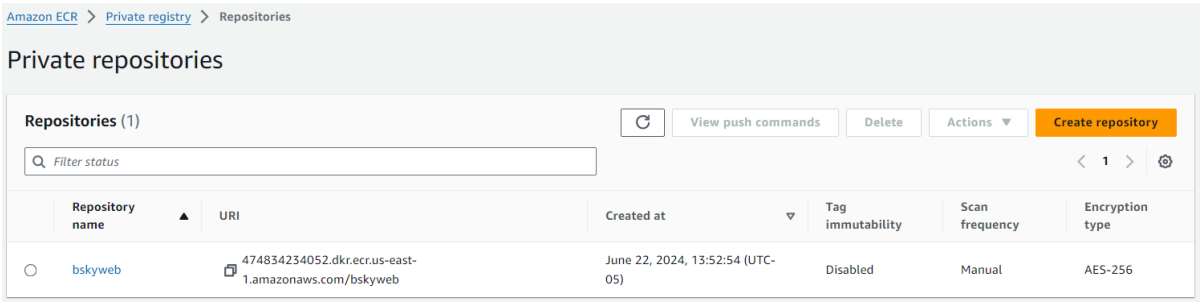

Figura 4: Creación de un repositorio privado en ECR

A partir de este punto, el flujo de trabajo denominado **build-and-push-bskyweb-aws** se ejecutará automáticamente cada vez que se realice un `push` a la rama 'main'. Verifica credenciales y el repositorio;

si es válido, ejecuta una serie de pasos en Ubuntu: checkout del código, configuración de Docker, login al registro, extracción de metadatos, configuración de outputs, y finalmente construcción y push de la imagen Docker. Si el repositorio no es válido, el proceso falla y muestra un error. El flujo termina con el despliegue exitoso de la imagen o con un fallo del workflow. La descripción de este flujo se ilustra en la Figura 5.

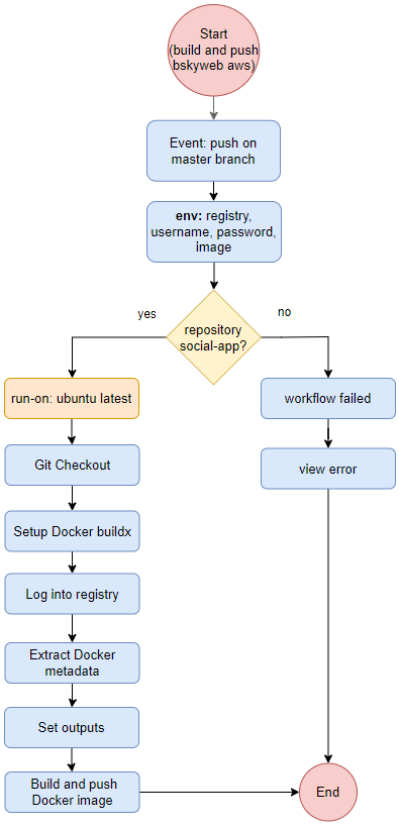

Figura 5: Workflow para la creación de la imagen bluesky

4.3. Configuración de Amazon EC2

Amazon EC2 es un servicio de AWS que proporciona capacidad de cómputo escalable en la nube. Permite crear y gestionar servidores virtuales. Se crearán dos instancias: una para `atproto` y otra para `social-app`.

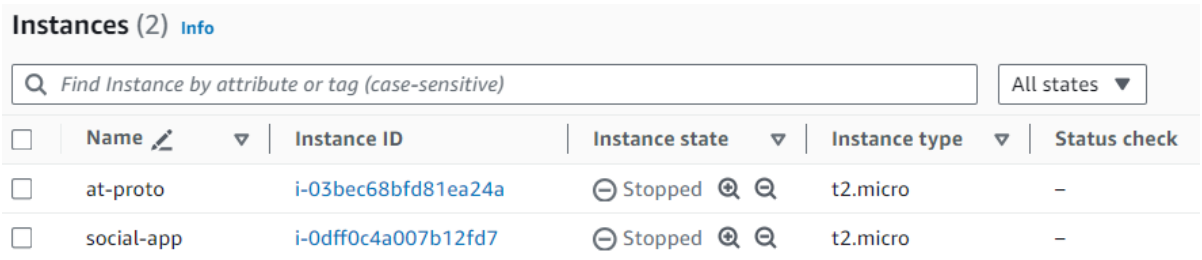

Figura 6: Creación de instancias

En la instancia atproto se deben ejecutar los siguientes comandos de construcción:

```
sudo apt install git jq golang docker
NVM=https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh
curl -o- $NVM | bash
sudo systemctl enable docker
sudo systemctl start docker
git clone https://github.com/bluesky-social/atproto
cd atproto
npm install --global pnpm
make nvm-setup
make deps
make build
make run-dev-env
```

En la instancia social-app, se deben ejecutar los siguientes comandos:

```
sudo apt install -y docker-ce
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker $USER
sudo docker pull public.ecr.aws/t8b2r8w9/social-app:latest
sudo docker run -d --name social-app -p 80:8100 public.ecr.aws/t8b2r8w9/social-app:latest /bin/sh -c "/usr/bin/bskyweb serve"
sudo iptables -A PREROUTING -t nat -i enX0 -p tcp --dport 80 -j REDIRECT --to-port 8100
```

4.4. Despliegue de la Aplicación

la Figura 7 muestra la arquitectura de la aplicación BlueSky alojada en AWS. Los usuarios acceden a la aplicación 'social-app' a través del puerto 80. Dentro de AWS, 'social-app' se comunica con 'at-proto' en el puerto 2583, que a su vez se conecta a un servicio externo llamado Render en el puerto 5432. Render aloja una base de datos PostgreSQL. El componente 'at-proto' también utiliza internamente el puerto 2581. Esta estructura permite la separación de la interfaz de usuario, la lógica de negocio y el almacenamiento de datos en diferentes servicios interconectados.

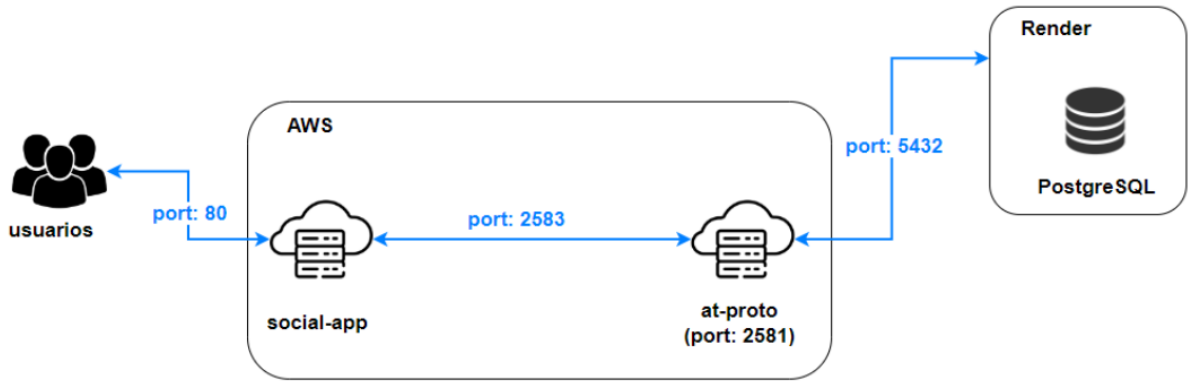

Figura 7: Despliegue en la nube en la Nube

5. Infraestructura Como Código

Otra forma automatizada de realizar el despliegue es aplicando Infraestructura Como Código usando Terraform.

Esto nos permitirá crear toda la infraestructura sin necesidad de realizar la creación de instancias EC2, grupos de seguridad o apertura de puertos de forma manual.

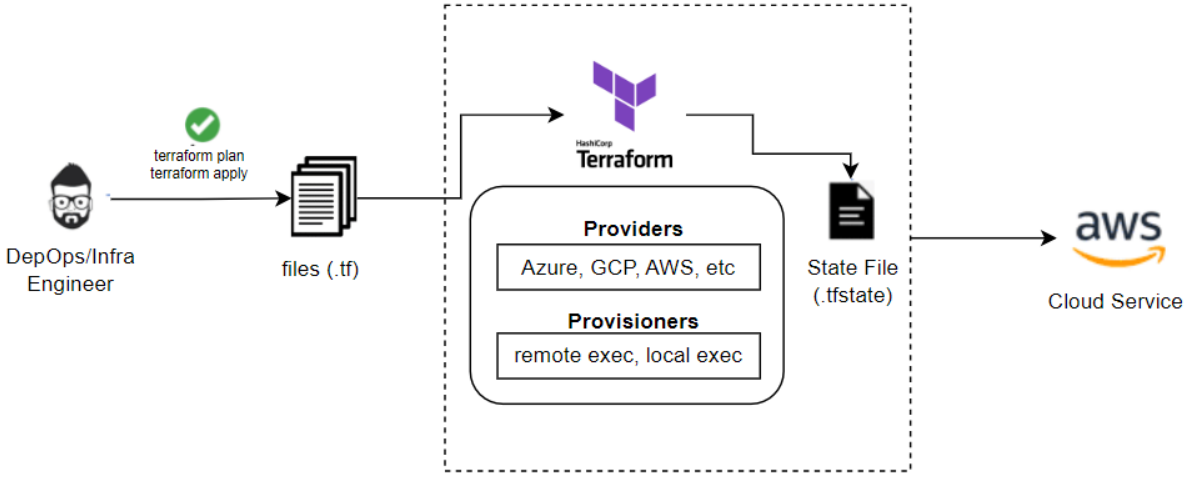

Figura 8: Infraestructura Como Código

Para llevar a cabo este procedimiento, se ha establecido un nuevo repositorio dedicado a Terraform. Este repositorio está estructurado en dos carpetas separadas: una destinada a `social-app` y otra a `atproto`.

Repositorio Terraform

https://github.com/richarquispe/bluesky-test

Es imprescindible tener instalado Terraform en su máquina local, ajustar las variables de usuario de AWS y ejecutar los comandos que se detallan a continuación:

```
terraform init
terraform plan
terraform apply
```

Estos comandos iniciarán el proceso de construcción de toda la infraestructura necesaria sin necesidad de acceder a la interfaz web de AWS.

6. Referencias

- Bluesky Social. Bluesky Social App [GitHub repository]. GitHub. Retrieved June 5, 2024, from https://github.com/bluesky-social/social-app
- Docker. Compose CLI reference. Docker Documentation. Retrieved June 5, 2024, from https://docs.docker.com/reference/cli/docker/compose
- Bluesky. AT Protocol. GitHub. Retrieved June 5, 2024, from https://github.com/bluesky-social/atproto
- Render. Retrieved July 11, 2024, from https://render.com/
- Amazon Web Services. Amazon Elastic Compute Cloud (EC2). Retrieved July 12, 2024, from ttps://docs.aws.amazon.com/ec2/
- Amazon Web Services. AWS Identity and Access Management (IAM). Retrieved July 12, 2024, from https://docs.aws.amazon.com/iam/
- Amazon Web Services. (n.d.). Amazon Elastic Container Registry (ECR). Retrieved July 12, 2024, from https://docs.aws.amazon.com/ecr/
- HashiCorp. Terraform Documentation. Retrieved July 12, 2024, from https://www.terraform.io/docs