Infraestructura Cloud y DevOps

DevOps y Arquitectura Escalable

Diseño de plataforma de video bajo demanda con infraestructura cloud moderna y segura

Contenido

Contexto Proyecto

Plataforma VOD

Desarrollar plataforma video bajo demanda similar a Netflix para usuarios globales

Requisitos Técnicos

Infraestructura escalable, segura y eficiente soportando miles de usuarios simultáneamente

Diseño Infraestructura Cloud

Modelo Implementación

AWS: La nube pública (AWS) permite elasticidad, pago por uso, alta disponibilidad global y servicios gestionados, lo que reduce la complejidad operativa. Esto es ideal para una startup o empresa en crecimiento que busca escalar rápido sin altos costos iniciales.

Servicios Cloud

IaaS (EC2): Para servicios como bases de datos en modo administrado (RDS).

PaaS (ECS/EKS): Para desplegar microservicios y manejar escalabilidad.

FaaS (Lambda): Para tareas como transcodificación o generación de thumbnails.

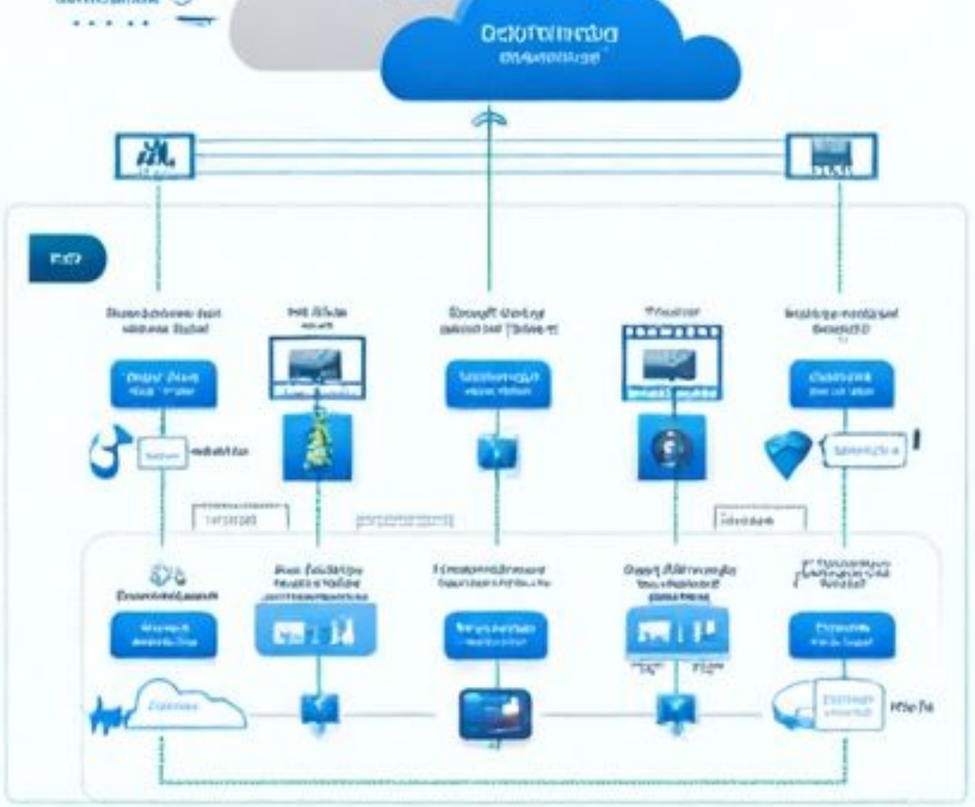
Almacenamiento Distribuido

Amazon S3: Para almacenar archivos de video.

Amazon Glacier: Para archivos de respaldo o históricos.

Amazon RDS: Para metadatos de usuarios y videos (PostgreSQL).

ElastiCache (Redis): Para sesiones de usuario y mejora de tiempos de respuesta.



Escalabilidad Horizontal

Auto Scaling

HPA (Horizontal Pod Autoscaler): en EKS según uso de CPU o peticiones.

Auto Scaling Groups: en caso de usar EC2 como worker nodes.

Load Balancing

Load Balancer (ELB): distribuye tráfico HTTP/S a múltiples pods.

Orquestación Contenedores AWS EKS

Kubernetes

Kubernetes ofrece portabilidad, escalabilidad automática, gestión avanzada de recursos y facilita CI/CD. Es ideal para un entorno con múltiples microservicios que deben escalar en demanda.

Docker

Contenedorización de aplicaciones y servicios distribuidos

Service Mesh

Comunicación segura entre microservicios y balanceadores

Registry

Gestión centralizada de imágenes de contenedores

Monitorización

Métricas

CloudWatch y Prometheus para seguimiento de rendimiento

Alertas

Sistema de notificaciones automáticas por umbrales

99.9%
Disponibilidad Sistema

< 2s
Latencia Streaming

10K+
Usuarios Concurrentes

Redes Seguras

- VPC con subredes públicas (para Load Balancer y frontend) y subredes privadas (para backend y base de datos).
- NAT Gateway para que recursos privados accedan a Internet.

Seguridad Avanzada

- IAM: acceso basado en roles.
- AWS Secrets Manager: para almacenar tokens/API Keys de forma segura.
- TLS/SSL para todo el tráfico de red.
- KMS: para cifrado en reposo (datos en RDS y S3).

CDN

Amazon CloudFront: entrega de video con baja latencia a nivel global, cacheando contenido en edge locations.

Edge Locations

Cache Inteligente

Geolocalización

Pipeline CI/CD

1

Código

Desarrollo y versionado con Git y GitHub Actions

2

Build

Construcción automática y tests de integración continua

3

Deploy

Despliegue automatizado con Terraform e IaC

4

Monitor

Monitorización continua con métricas y logging

5

Feedback Loop

Retroalimentación para mejora continua del proceso

Pipeline CI/CD Código

Github actions

deploy.yml

```yaml
name: CI/CD - VOD Platform

on:
  push:
    branches:
      - main  # Ejecuta solo en rama main
  pull_request:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
    # 1. Clona el repositorio
    - name: Checkout code
      uses: actions/checkout@v3

    # 2. Configura credenciales de AWS (requiere secretos configurados)
    - name: Configure AWS Credentials
      uses: aws-actions/configure-aws-credentials@v2
      with:
        aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
        aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
        aws-region: us-east-1

    # 3. Configura Terraform CLI
    - name: Set up Terraform
      uses: hashicorp/setup-terraform@v2
      with:
        terraform_version: 1.5.5

    # 4. Inicializa Terraform (descarga plugins, prepara estado)
    - name: Terraform Init
      run: terraform init

    # 5. Planifica cambios en la infraestructura
    - name: Terraform Plan
      run: terraform plan

    # 6. Aplica cambios automáticamente
    - name: Terraform Apply
      run: terraform apply -auto-approve

    # 7. Login a Amazon ECR (repositorio de contenedores)
    - name: Login to Amazon ECR
      uses: aws-actions/amazon-ecr-login@v1

    # 8. Construye y sube la imagen Docker del backend
    - name: Build and Push Docker Image
      run: |
        IMAGE_URI="${{ secrets.AWS_ACCOUNT_ID }}.dkr.ecr.us-east-1.amazonaws.com/vod-backend:latest"
        docker build -t vod-backend .
        docker tag vod-backend:latest $IMAGE_URI
        docker push $IMAGE_URI

    # 9. Configura acceso al cluster EKS
    - name: Configure kubeconfig
      run: |
        aws eks update-kubeconfig --name vod-cluster --region us-east-1

    # 10. Despliega los manifiestos de Kubernetes (k8s/)
    - name: Deploy to EKS
      run: |
        kubectl apply -f k8s/
```

Pipeline CI/CD Código

Terraform

main.tf

main.tf > module "eks" > enable_irsa

```hcl
# -------- providers.tf --------
# Define el proveedor y región a utilizar
provider "aws" {
  region = "us-east-1"
}

# -------- variables.tf --------
# Variables reutilizables para entorno
variable "region" {
  default = "us-east-1"
}

variable "cluster_name" {
  default = "vod-cluster"
}

variable "vpc_id" {
  description = "ID de la VPC existente"
}

variable "subnet_ids" {
  type = list(string)
  description = "Lista de subredes para los nodos EKS"
}

# -------- s3.tf --------
# S3 para almacenamiento de videos
resource "aws_s3_bucket" "vod_videos" {
  bucket = "vod-video-storage-bucket"

  tags = {
    Name = "VideoStorage"
    Project = "VOD"
  }
}

# Versionado para manejar múltiples versiones de videos
resource "aws_s3_bucket_versioning" "versioning" {
  bucket = aws_s3_bucket.vod_videos.id

  versioning_configuration {
    status = "Enabled"
  }
}

# Cifrado del bucket con AES256 (por defecto)
resource "aws_s3_bucket_server_side_encryption_configuration" "encryption" {
  bucket = aws_s3_bucket.vod_videos.id

  rule {
    apply_server_side_encryption_by_default {
      sse_algorithm = "AES256"
    }
  }
}

# -------- eks.tf --------
# Crea el cluster EKS utilizando un módulo oficial de Terraform
module "eks" {
  source          = "terraform-aws-modules/eks/aws"
  cluster_name    = var.cluster_name
  version         = "1.27"
  subnets         = var.subnet_ids
  vpc_id          = var.vpc_id

  # Habilita la autenticación IRSA
  # (IAM Roles for Service Accounts)
  enable_irsa     = true

  # Define un grupo de nodos (EC2) para ejecutar los pods
  node_groups = {
    default = {
      desired_capacity = 2
      max_capacity     = 4
      min_capacity     = 1
      instance_type    = "t3.medium"

      labels = {
        role = "general"
      }
    }
  }

  tags = {
    Environment = "dev"
    Project     = "VOD"
  }
}
```

Herramientas y Tecnologías Principales

Cloud Providers

- AWS con EC2, S3, CloudFront
- Servicios serverless Lambda/Functions

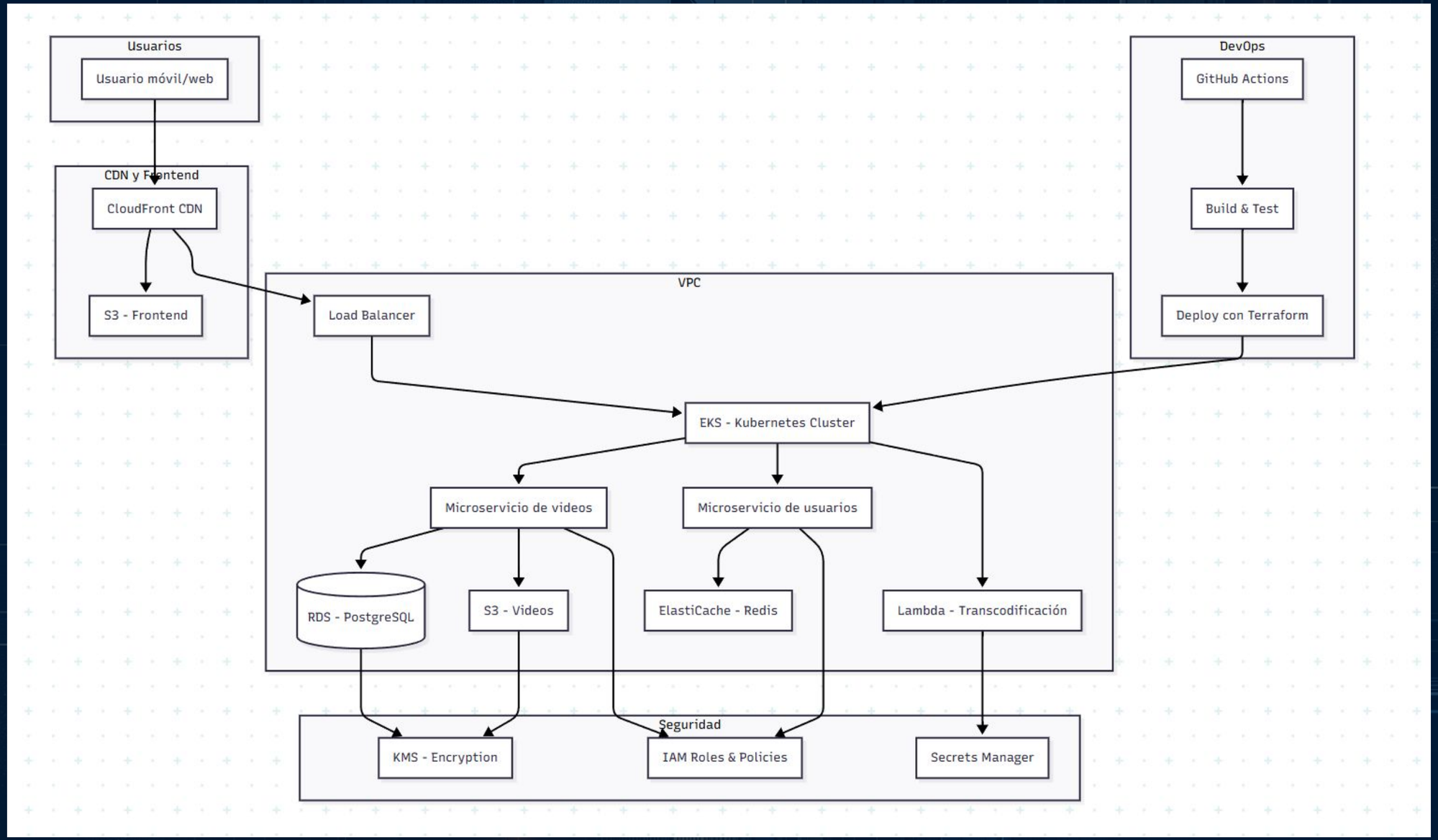
DevOps Tools

- Docker para contenedorización
- Kubernetes para orquestación
- Terraform para IaC
- GitHub Actions para CI/CD

Monitorización

- CloudWatch para métricas AWS
- Prometheus y Grafana stack
- ELK Stack para logging
- Alertmanager para notificaciones

Diagrama Arquitectura



Infraestructura Cloud VOD - Proyecto DevOps

Diseño completo de arquitectura escalable para plataforma de video bajo demanda empresarial

https://github.com/CesarHerr/Ev_devOps_Adalid/tree/main/modulo5