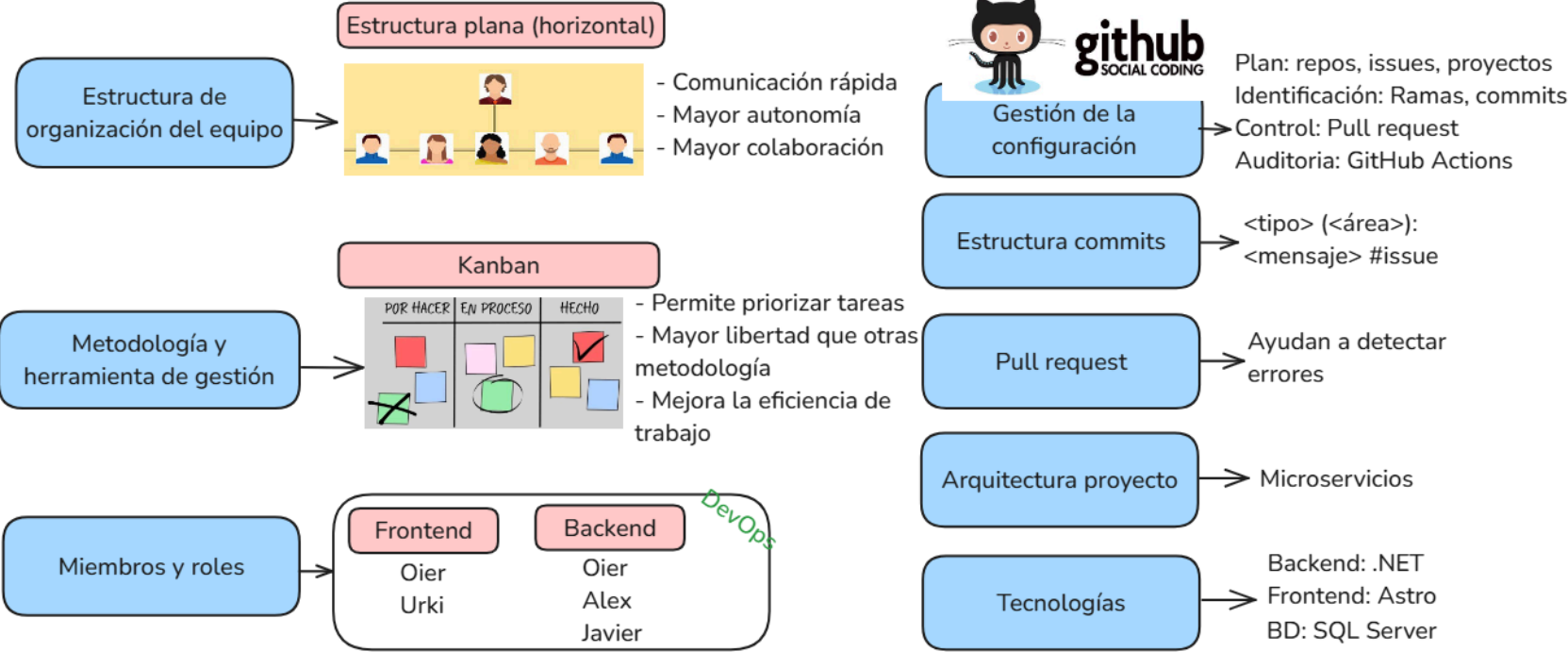


RFP - Proyecto Votación

Autores: Oier A., Urki A., Oier L., Javier P., Álex S.

RFI I. Proyecto Votación - Metodologías de Gestión

Resumen



RFI II. Proyecto Votación - Arquitectura Basada en Microservicios

MICROSERVICIOS y FUNCIONALIDAD

Candidatos



- Gestión de la información relacionada con los candidatos
 - * Información personal
 - * Votos
- Desarrollo de Web API (CRUD)

GET /api/Candidates/ObtenerCandidatoPorId/{id}

POST /api/Candidates/InsertarNuevoCandidato

PUT /api/Candidates/ActualizarVotosCandidato

DELETE /api/Candidates/EliminarCandidato/{id}

Autenticación y autorización



- JWT: forma segura de compartir información
- Necesario para consultar la API Candidatos
- Genera token con 1 hora de validez
- Privilegios de administrador y normal
- Contraseñas - Base64 - hashear SHA-512

Authorization

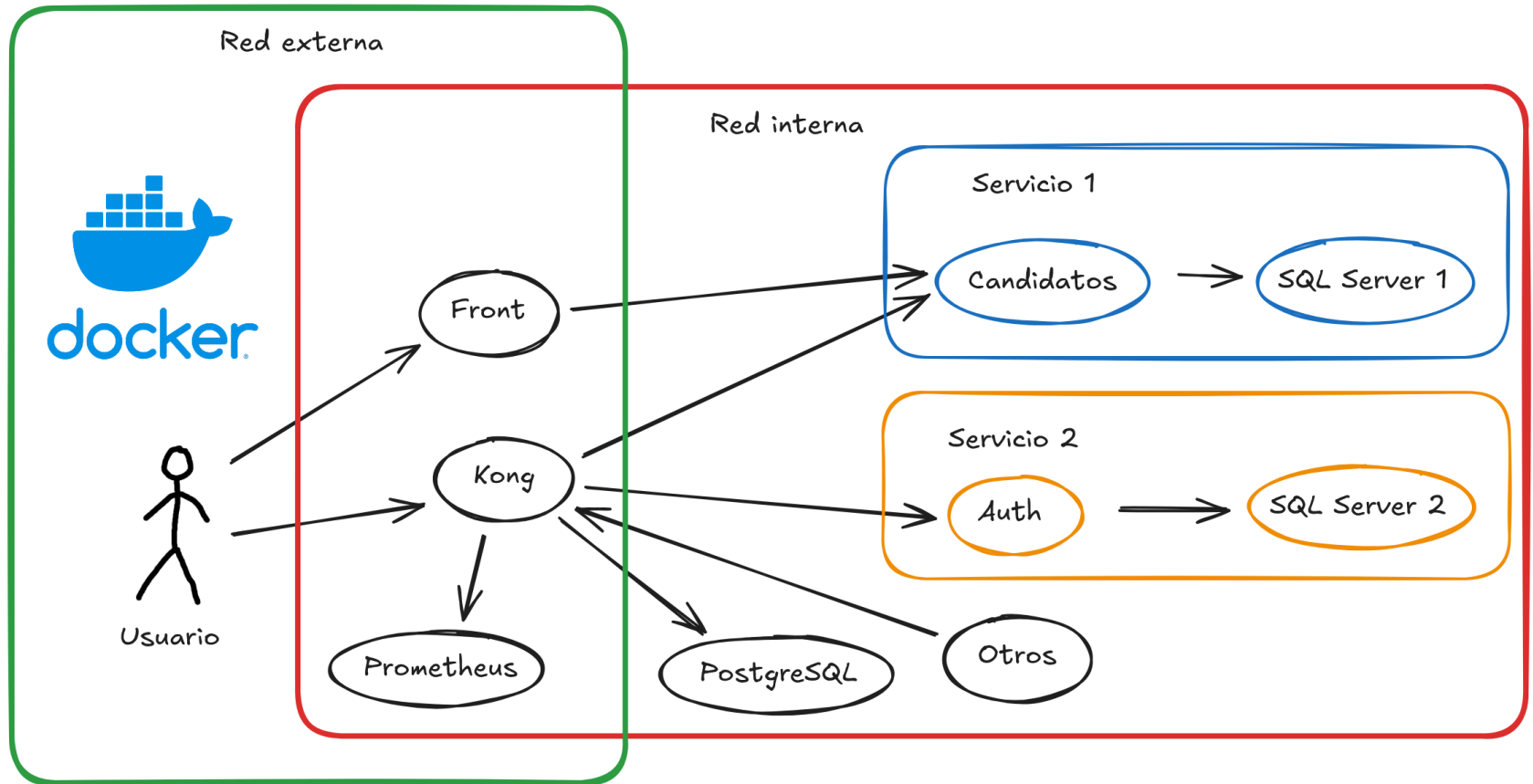
POST /api/Auth/Login

POST /api/Auth/Register

☒ Usuarios

☒ Comentarios

ARQUITECTURA Y COMUNICACIÓN ENTRE



DESPLIEGUE EN ENTORNO LOCAL

CONTINUACIÓN DEL RFI I



Una vez dockerizado el proyecto, el despliegue se se lleva a cabo haciendo uso de docker-compose.

Se despliegan de este modo todos los contenedores a la vez.

PASOS PARA DESPLIEGUE EN LOCAL USANDO DOCKER-COMPOSE

1. Situar .env en raíz del proyecto
2. `docker-compose up -d kong-migration`
3. `docker-compose up -d --build`
4. `docker-compose up -d kong-config`

} Ejecutar una única vez

Para detener los contenedores: `docker-compose down`

Para ejecutar de nuevo los contenedores: `docker-compose up -d`

AUTORIZACIÓN, AUTENTICACIÓN Y AUD.

AUTENTICACIÓN

- Empleo de servicio de Autenticación para obtener token
- Verificación de usuario y contraseña
- Token validez 1 hora
- JWT: Estándar para autenticación
- Rol: Usuario normal



AUTORIZACIÓN

- Kong (Api Gateway) valida token
- Rol Admin: Permite todas las llamadas
- Rol Normal: Permite solo consultas



AUDITORIA

- Plugin "file-log" para guardar logs de consultas
- Se crea uno por cada servicio
- Ubicación /tmp
- Kong tiene su propia BD



SOLUCIÓN ESCALABLE Y ESLÁSTICA

✓ Arquitectura escalable y elástica

Enfoque por capas:

Frontend, APIs y base de datos escalan de forma independiente

Servicios desacoplados:

Escalado horizontal o vertical según la demanda

Uso de contenedores:

Docker facilita despliegues portables y gestionados con Docker Compose

-> Aporta escalabilidad

-> Aporta elasticidad

CI/CD con GitHub Actions:

Integración y despliegue continuo para adaptabilidad

-> Se ajusta a nuevas demandas

◆ Resultado: Solución flexible, eficiente y preparada para crecimiento

RFI III - RFP. Proyecto Votación - CI/CD & Arquitectura Cloud

Servicios necesarios



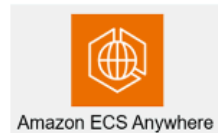
Amazon Elastic Container Registry (Amazon ECR)

2 ~ 0.6€ / mes



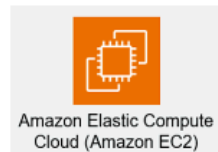
Amazon Elastic Container Service (Amazon ECS)

1 En EC2



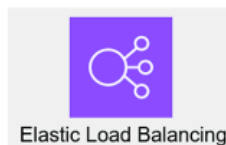
Amazon ECS Anywhere

2 En EC2



Amazon Elastic Compute Cloud (Amazon EC2)

4 ~ 28€ / mes



Elastic Load Balancing

1 ~ 16€ / mes



Amazon Relational Database Service (Amazon RDS)

1 ~ 25€ / mes



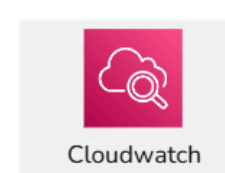
Security Group

2 ~ Gratis 🇪🇺



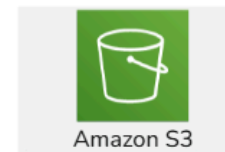
Amazon API Gateway

1 ~ 3€ / mes



Cloudwatch

1 ~ 2€ / mes



Amazon S3

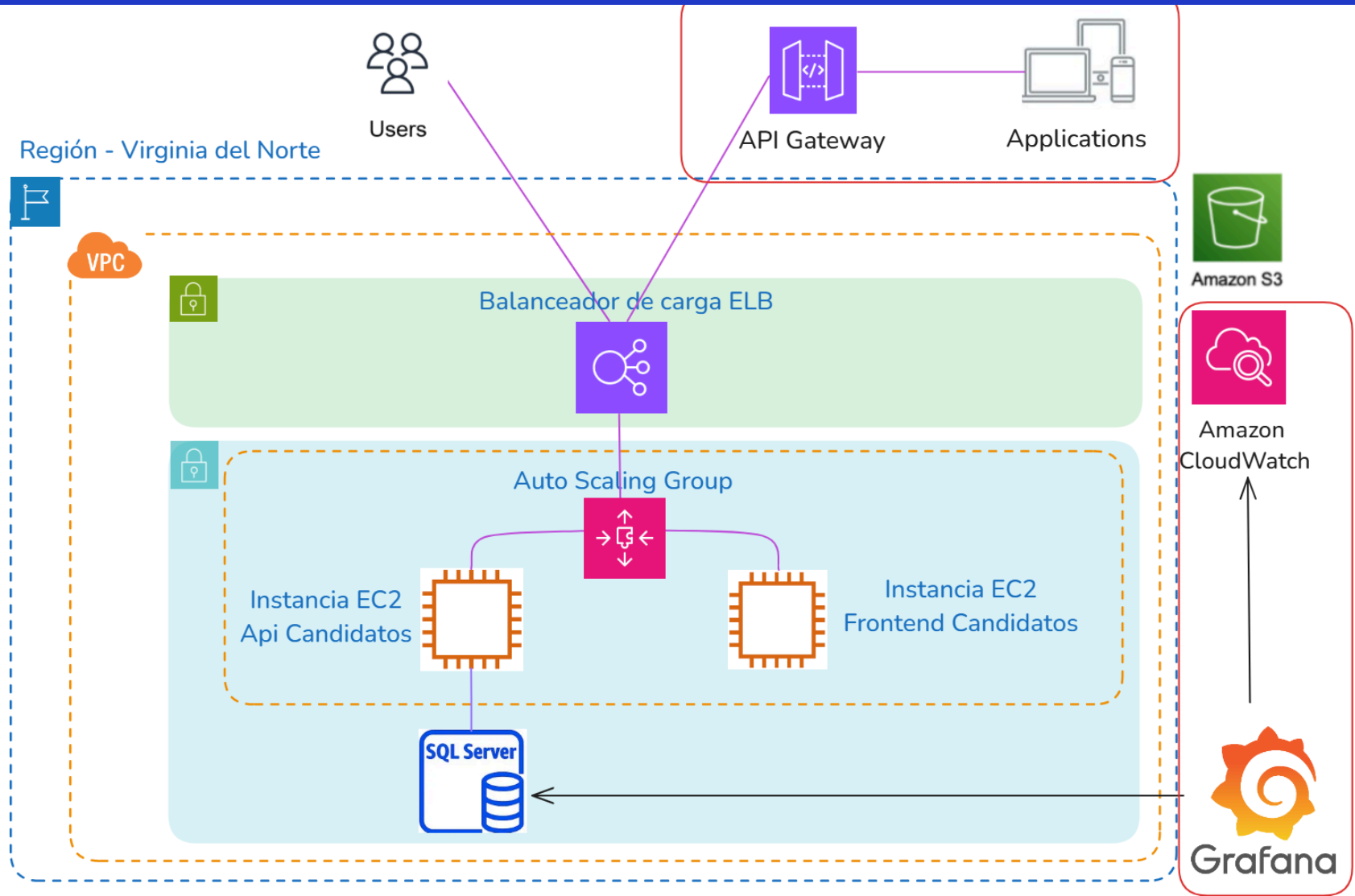
2 ~ 0.6€ / mes



Secrets Manager

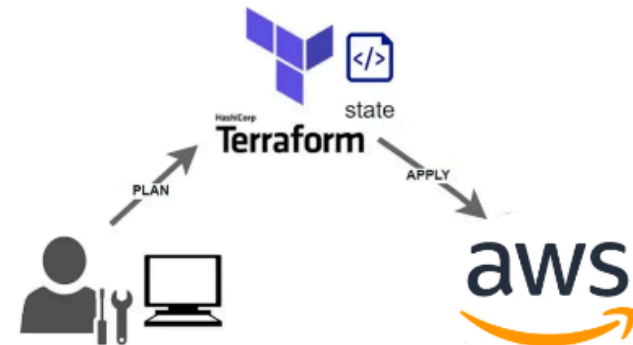
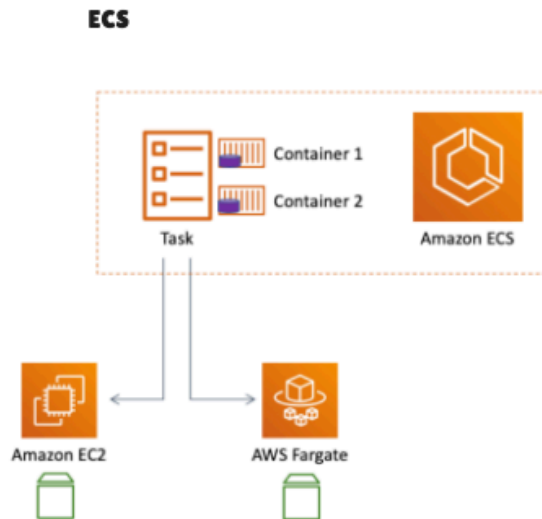
2 ~ 0.8€ / mes

Arquitectura AWS

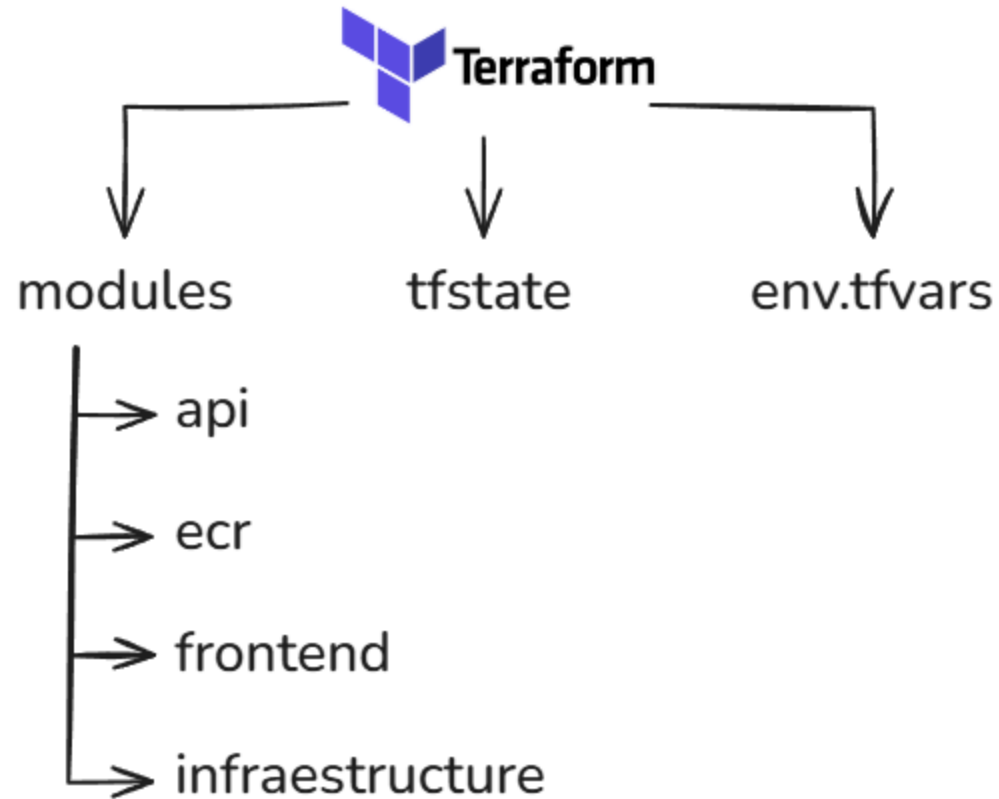


Escalabilidad y elasticidad

Escalabilidad y Flexibilidad



Terraform



✓ Backend - Ejecutar StyleCop Linter

✓ Frontend - Ejecutar ESLint

✓ Backend - Ejecutar Tests

✓ Frontend - Ejecutar Tests



Integración
continua

✓ Terraform - Crear ECR

✓ Build, Tag y Push imágenes a ECR

✓ Terraform - Crear infraestructura

✓ Poblar base de datos

✓ Infracost - Cálculo de costes



Despliegue
continuo


Metricas / logs




Amazon Cloudwatch




Infracost - Pull Request

 **infracost** bot commented 2 days ago • edited

 **Infracost report**

Consider fixing these issues, they don't align with your company's FinOps policies & the Well-Architected Framework. Add a PR comment with `@infracost help` to see how you can dismiss or snooze issues and unblock your PR.


Tagging policies

►  FinOps tags

resource `module.api.aws_ecs_service.service_api_candidatos`

- Missing mandatory tag `Environment`. Must be one of `dev`, `stage`, `prod`. Consider using default tags to avoid adding tags to individual resources.
- Missing mandatory tag: `Service`. Consider using default tags to avoid adding tags to individual resources.
- Dynamically created `task` resources will not have tag(s) `Service`, `Environment` because tag propagation is not configured. Tag propagation should be configured by setting `propagate_tags` to a valid value (`TASK_DEFINITION`, `SERVICE`)

in project `Terraform-terraform`

▼ Monthly estimate decreased by \$10 

Changed project	Baseline cost	Usage cost*	Total change	New monthly cost
Terraform-terraform	-\$10	-	-\$10 (-13%)	\$69

*Usage costs can be estimated by updating [Infracost Cloud settings](#), see [docs](#) for other options.

► Estimate details

Infracost - Cloud

FinOps policies 7 issues

● ECS - consider using Graviton instances 2 issues ▾

● ECR - consider using a lifecycle policy 2 issues ▾

● Cloudwatch - consider using a retention policy to reduce storage costs 1 issue ▾


● EC2 - consider using latest generation instances for t family instances 1 issue ▲

resource **module.infrastructure.aws_autoscaling_group.AutoScalingAutoScalingGroup**

- Switch `launch_template.instance_type` from `t2.micro` to `t3.micro` — save \$10/year (9%)
in project `oielay/GTIO_Votacion/Terraform`

● EC2 - consider using Graviton instances 1 issue ▾

Cost estimate

 **\$69** monthly cost ⓘ

1. Solución Cloud

a. Entorno de desarrollo

- EC2 (instancias Api y Frontend): ~2 x t2.micro → ~\$17/mes
- RDS (db.t3.micro, bajo uso): ~\$18/mes
- ELB: ~\$16/mes (uso compartido)
- Costes totales estimados: ~\$51/mes → ~45€/mes

b. Entorno de producción

- EC2 (Api, Frontend y Grafana): ~5 x t2.micro → ~\$42.5/mes
- RDS : ~\$18/mes
- ELB : ~\$16/mes
- CloudWatch logs (según retención): \$0–2/mes
- Total estimado: ~\$75–80/mes → ~\$66–70/mes

c. Entornos de desarrollo locales

- Sin coste en AWS
- Uso de Docker

Costes

2. Costes de CI/CD

- Uso de GitHub Actions gratuito
- Uso de Infracost gratuito (cuenta Cloud con límite)
- Almacenamiento de `.env` y `env.tfvars` en S3 (mínimo coste mensual)
- Estimación total: ~1–2€/mes (solo S3 y posibles logs)

3. Costes RRHH

Rol	Nº Personas	Sueldo medio	Dedicación estmiada	Coste estimado
Desarrollador junior	5	20€/h	110h	11.000€

4. Otros costes (licencias e infraestructuras)

- Docker Desktop (uso gratuito en educación)
- Terraform CLI (open source)
- Infracost (open source + cuenta gratuita)
- VS Code / IDEs (gratuitos)
- GitHub (gratuito)
- Total estimado: \$0

Costes

Total

Aspecto	Coste mensual	Coste Anual
Entorno Desarrollo	45€/mes	540€/año
Entorno Produccion	70€/mes	840€/año
Entorno Desarrollo Local	0€/mes	0€/año
CI/CD	2€/mes	24€/año
Total	117€/mes	1404€/año

RRHH	Horas/trabajador	Coste hora	Coste Total
5 prog. jr.	110h	20€/h	11.000€



DEMO