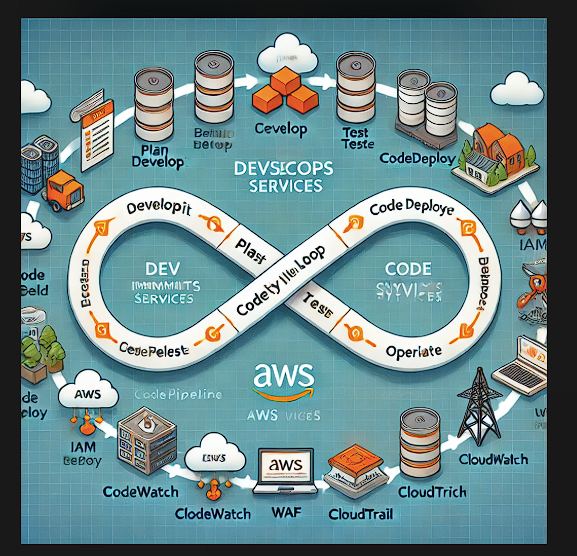
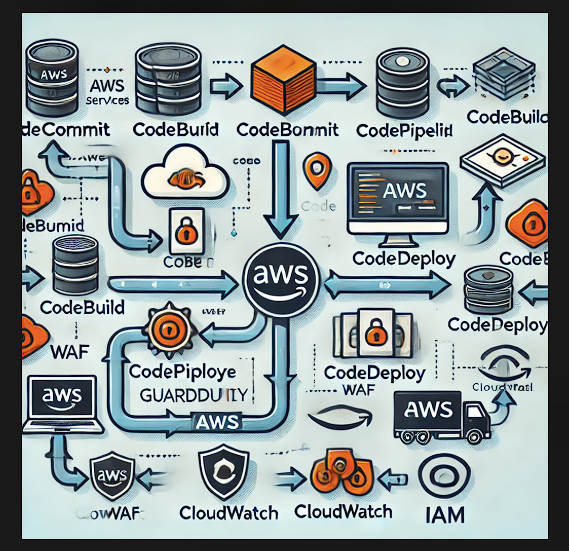
**Temas a tratar, como propuesta de migración Legacy, a DevSecOps, sobre AWS.**



|  |  |  |
| --- | --- | --- |
| **Fase** | **Herramientas.** | |
| **Legacy** | **AWS** |
| **Planificación** | **GitLab** | **AWS IAM** |
| **Desarrollo** | **SonarQube GitLab Snyk Trivy** | **Amazon CodeGuru Amazon Inspector** |
| **Integración** | Jenkins (**SonarQube Snyk Trivy**) | **AWS CodePipeline AWS CodeBuild AWS Lambda** |
| **Despliegue** | Jenkins | **AWS CodeDeploy AWS Elastic Beanstalk Amazon ECS Amazon EKS**  **AWS CloudFormation** |
| **Operación** | **SonarQube Snyk Trivy** | **Amazon CloudWatch AWS CloudTrail AWS Config AWS Security Hub Amazon GuardDuty AWS WAF AWS Transit Gateway** |
| **Retroalimentación** | **GitLab** | **AWS CloudWatch Logs AWS X-Ray AWS Security Hub** |

**Herramientas AWS**:

* **Amazon CodeGuru**: Para análisis de código y detección de vulnerabilidades de seguridad.
* **CodeGuru Reviewer**: Analiza el código fuente en busca de errores comunes, vulnerabilidades de seguridad y malas prácticas de programación. Integra con servicios de repositorios como **GitHub** y **CodeCommit**.
* **CodeGuru Profiler**: Proporciona recomendaciones sobre el rendimiento de las aplicaciones, ayudando a identificar cuellos de botella y optimizando el uso de los recursos.
* **AWS Lambda**: Para la ejecución de funciones que pueden formar parte de la integración.
* **AWS X-Ray**: Para analizar y depurar el rendimiento de las aplicaciones.

**Integración:**



* **Desarrollo de Pipelines:** Traducir las etapas de **Jenkinsfile** a **YAML** en **AWS CodePipeline**, utilizando los servicios de AWS adecuados para cada fase (**EKS**, etc.).
* **Snyk** y **Trivy** se pueden integrar en **CodePipeline**, para realizar escaneos de seguridad en el código y en las imágenes de contenedores durante las fases de integración y despliegue. También tienen capacidades para detectar vulnerabilidades en los entornos de producción, por lo que pueden ser utilizadas en la fase de **Operación**.
* Con **CloudFormation** añadido en la fase de despliegue, se refuerza el soporte para **infraestructura como código**, permitiendo una gestión más estructurada y automatizada de los recursos en AWS.

**Cumplimiento**:

* **Seguridad Integral:** Con servicios como **Security Hub**, e **Inspector**, se logra un monitoreo constante de las amenazas, mitigación de riesgos, y protección a nivel de infraestructura y aplicaciones.
* **Monitoreo y Visibilidad:** Gracias a **CloudWatch** y **CloudTrail**, se proporciona visibilidad completa sobre el estado y las actividades de la infraestructura, lo cual es clave para la respuesta ante incidentes.
* Integrar con **ALB**, o **CloudFront** para proteger aplicaciones desplegadas.
* Monitorear logs de seguridad de **WAF**, **VPC** **Flow** **Logs** o eventos detectados.
* Usar **CloudWatch Logs Insights** para analizar errores en la compilación o despliegue.
* Centralizar hallazgos de herramientas como **GuardDuty**, **AWS Config** y **Amazon Inspector** para ofrecer un panorama unificado de los riesgos.
* Evaluar la conformidad con estándares de seguridad como **CIS AWS Foundations Benchmark** o normativas específicas del cliente.
* **Automatización de despliegue**: Código desplegado desde el repositorio, pasa por las fases de construcción, pruebas, seguridad, y finalmente el despliegue a ambientes de producción o staging (a través de **Elastic Beanstalk**, **ECS** o **EC2**).
* Los artefactos construidos necesitan ser desplegados en múltiples regiones de AWS o en diferentes **VPCs,** utilizando **Transit Gateway,** facilita esta comunicación.

**Gobernanza**:

* Integrar **IAM** y **Security Groups** con **CodePipeline**, **CodeBuild** y **CodeDeploy** para asegurar que cada etapa del pipeline se ejecute solo con los permisos adecuados.
* **GitLab** y **SonarQube** como parte de las herramientas existentes de la continuidad del negocio, la integración de estos en el flujo específico de **AWS CodePipeline** debe ser detallada para asegurar que los artefactos y el análisis de código sigan el flujo de DevSecOps dentro de AWS.
* Implementar políticas automatizadas contra ataques DDoS con **AWS Shield Advanced**.
* Detectar anomalías en el tráfico de red o acceso sospechoso a servicios (como intentos de acceso no autorizados a instancias, o buckets de **S3**)..

**Itinerario.**

**Semana 1.**

* **Intensidad: 4 Horas diarias.**
* **Horario: lunes a sábado, las dos semanas.**
* **Temas: Separados en bloques de color por fase de servicio.**

|  |  |  |  |
| --- | --- | --- | --- |
| **AWS.** | **Temas.** | **Herramienta.** | **Script.** |
| **Gobernanza** | **Configuración de la CLI de AWS en local.** | **IAM.** | **login\_aws.sh** |
| **S3.** | **s3.sh** |
| **Crear métricas personalizadas y alertas de seguridad.** | **AWS Backup.** |  |
| **CI/CD.** | **Escribir un Jenkinsfile simple, convertirlo a YAML y validar la salida.** | **Python.** | **migrate\_jenkins\_to\_codepipeline.py** |
| **Integración con pipelines.** | **CodePipeline.** | **yaml.sh** |
| **Configuración de eventos para desencadenar.** | **GitLab.** |  |
| **Exploración.** | **CodeBuild.** |  |
| **Automatización.** | **CodeDeploy.** |  |
| **Crear y actualizar plantillas avanzadas con parámetros dinámicos y salidas.** | **CloudFormation.** | **codepipeline.yml** |
| **Verificar cambios en una pila existente.** |
| **Conexión.** | **EC2.** | **ec2.yml** |
| **RDS.** |  |
| **Lambda.** |  |
| **Networking.** | **Configurar.** | **API Gateway.** |  |
| **Configurar.** | **CloudFront.** |  |
| **Contenedores.** | **Configurar balanceo de carga.** | **ALB.** |  |
| **Configurar auto-scaling.** | **ECS.** | **ecs.yml** |
| **Subir imágenes generadas por un pipeline CI/CD.** | **ECR.** | **ecr.yml** |
| **Desplegar una aplicación con Fargate desde ECR.** | **Fargate.** |  |
| **Despliegue simplificado de aplicaciones web y contenedores.** | **Elastic Benstalk.** |  |
| **Desplegar una aplicación de múltiples contenedores.** | **EKS.** | **eks.yml** |
| **Orquestación.** | **Namespaces.** |
| **RBAC.** |
| **Ingress controllers.** |

**Semana 2.**

* **Nota: DevSecOps: Ocupa dos (2) dias (d) del programa.**
* **Total: 48 Horas, las dos semanas.**
* **Publico: Hasta 10 personas en reunión.**

|  |  |  |  |
| --- | --- | --- | --- |
| **AWS.** | **Temas.** | **Herramienta.** | **Script.** |
| **Seguridad.** | **Crear y consumir secretos en una aplicación.** | **Secrets Manager.** |  |
| **Configurar reglas de conformidad para monitorear recursos.** | **AWS Config.** |  |
| **Escaneo de vulnerabilidades para instancias.** | **AWS Inspector.** |  |
| **Crear y aplicar reglas básicas para proteger aplicaciones desplegadas.** | **AWS WAF.** | **waf.sh** |
| **Crear métricas personalizadas y alertas de seguridad.** | **AWS CloudWatch.** |  |
| **Crear métricas personalizadas y alertas de seguridad.** | **AWS Backup.** |  |
| **Observabilidad.** | **Configurar una traza para una aplicación web.** | **X-Ray.** |  |
| **Generar y revisar logs para auditoría de acceso a recursos.** | **AWS CloudTrail.** |  |
| **Uso para notificaciones de eventos.** | **SNS.** |  |
| **Uso como sistema de mensajería confiable.** | **SQS.** |  |
| **DevSecOps.** | **Integrar y probar las herramientas de seguridad en un pipeline para monitoreo continuo.** | **SonarQube.** |  |
| **CodeGuru.** |  |
| **Escanear dependencias de un proyecto y mostrar vulnerabilidades encontradas.** | **Snyk.** |  |
| **Realizar un análisis de seguridad en una imagen.** | **Docker.** |  |
| **Trivy.** |  |
| **Hub Security.** |  |
| **Mejora Continua.** | **Analizar las alertas generadas.** | **GuardDuty.** |  |
| **Mejoras Continuas con CI/CD y Seguridad.** | **Laboratorios.** |  |
| **Troubleshooting.** | **Recapitulación de la adopción de herramientas de seguridad en AWS.** |
| **Importancia de la integración de DevSecOps para un flujo de trabajo más seguro y automatizado.** |
| **Tareas y recomendaciones para continuar con la integración de seguridad en producción.** |