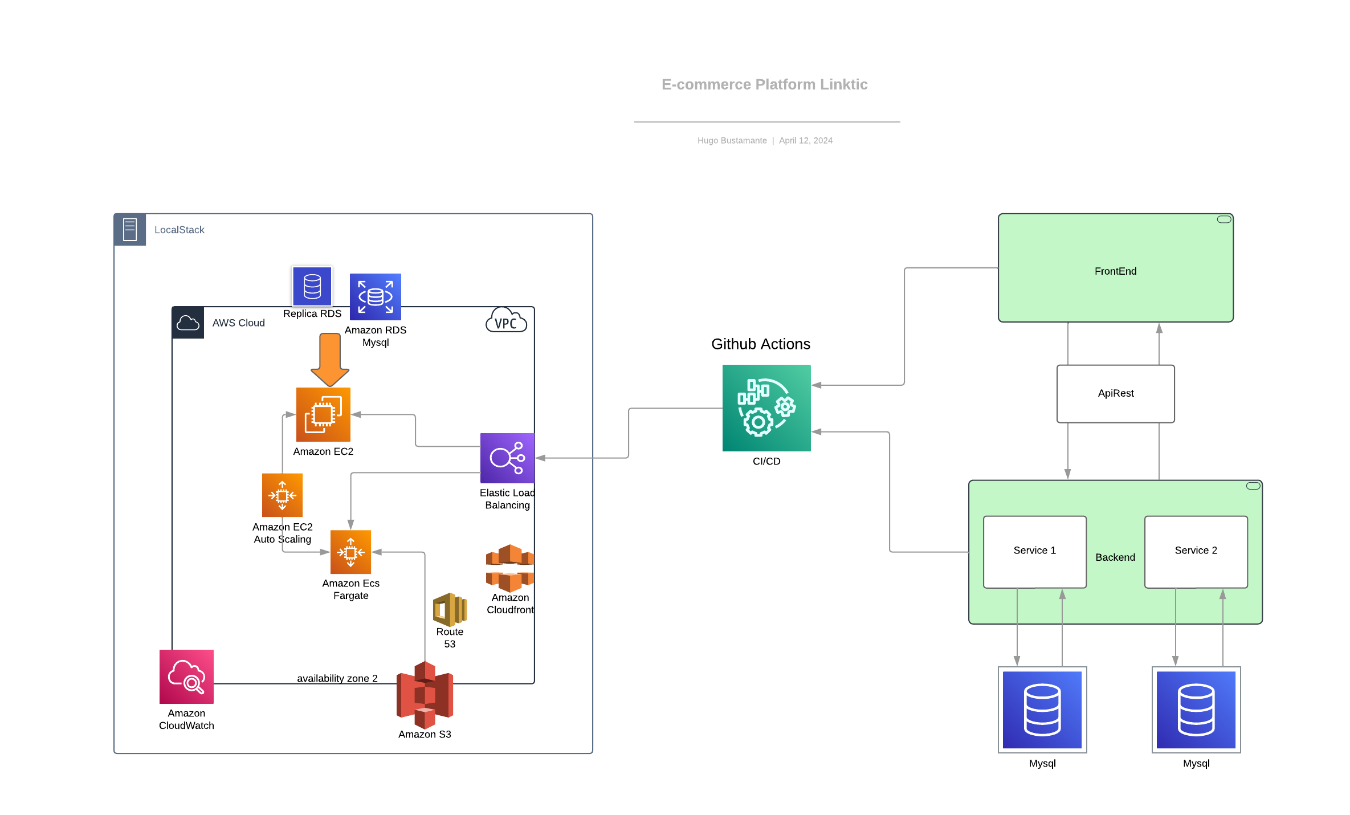
**Arquitectura de software**

Para el desarrollo local de este proyecto utilice la arquitectura por microservicios. Para este desarrollo se implemento dos servicios en el backend con dos lenguajes de programación, estos son Java y Nodejs dedicados a gestionar el catalogo de productos y gestionar los pedidos realizados en el ecommerce, cada servicio backend tiene su propia base de datos con el fin de tener desacoplamiento, autonomía, escalabilidad y rendimiento. Junto con el Frontend hecho en Angular, estos 3 servicios se estaran comunicado por medio de un servicio API RestFull. Una vez concluido el desarrollo del codigo se implementa un flujo de CI utilizando herramientas como GitHub Actions, cada vez que se realiza un push al repositorio, se activaba automáticamente un conjunto de pruebas automatizadas para garantizar que los cambios no afectaran la estabilidad del sistema.

Una vez que el código pase exitosamente por el proceso de CI, es el momento de desplegar la aplicación en un entorno local para realizar pruebas más exhaustivas. Utilizamos LocalStack, una herramienta que nos permite simular servicios de la nube de AWS en un entorno local. Podriamos Configurar instancias EC2, load balancers, VPC, S3 para guardar archivos estaticos de la web y bases de datos como Amazon RDS, todo en LocalStack para replicar fielmente nuestro entorno de producción. Para este desarrollo del flujo normal mencionado anteriormente, no llegamos a usar S3, EC2, RDS, Cloudfront ningun servicio de despliegue que ofrece aws en LocalStorage para realizar el despliegue de nuestra app, sin embargo dejo un diagrama de arquitectura sobre lo que se queria implementar para todo el ciclo de vida del desarrollo de la app. Del siguiente diagrama de flujo se llego a implementar hasta la etapa del CICD:



A continuación planteare una arquitectura para el E-commerce en un ambiente real ya sea de producción o desarrolloro:

Para diseñar una arquitectura que cumpla con los requisitos y alineamientos de nuestra E-commerce Linktic en un entorno de despliegue en producción o desarrollo, podemos considerar lo siguiente:

**Arquitectura Propuesta:**

1. Descripción General:

* Se utilizará una arquitectura de microservicios para modularizar la aplicación y permitir el desarrollo y despliegue independiente de cada servicio.
* Se implementará un proceso de CI/CD para automatizar la integración, pruebas y despliegue de los servicios.
* Se utilizarán servicios de AWS para la infraestructura en la nube, garantizando escalabilidad, disponibilidad y seguridad.

2. Servicios y Componentes:

**2.1. Frontend (Angular, React):**

* Se desarrollará como una aplicación web estática y se hospedará en Amazon S3.
* Se utilizará CloudFront como CDN para mejorar la entrega de contenido estático y la escalabilidad.

**2.2. Backend de Gestión de Catálogos (Java, c#, Python, Nodejs):**

* Se desarrollará como un servicio RESTful utilizando Spring Boot.
* Se desplegará en instancias EC2 en una VPC privada para garantizar la seguridad.
* Se configurará un Load Balancer de Application para distribuir el tráfico entre las instancias EC2.

**2.3. Backend de Gestión de Pedidos (Node.js, Java, C#, Python):**

* Se desarrollará como un servicio RESTful utilizando Express.js.
* Se desplegará en instancias EC2 en una VPC privada.
* Se configurará un Load Balancer de Application para distribuir el tráfico entre las instancias EC2.

**2.4. Base de Datos:**

* Se utilizará Amazon RDS para alojar la base de datos relacional (por ejemplo, MySQL) para almacenar información de catálogos y pedidos.
* Se implementará en una VPC privada para limitar el acceso desde el exterior y se configurará la seguridad a nivel de grupo de seguridad de AWS.

**2.5. Herramientas de CI/CD:**

* Se utilizará GitHub como repositorio de código para gestionar versiones y colaboración.
* Jenkins se empleará como servidor de automatización para orquestar el flujo de trabajo de CI/CD.
* SonarQube se utilizará para realizar análisis estáticos de código y mejorar la calidad del código.
* Ansible se empleará para la gestión de la configuración y el aprovisionamiento de infraestructura.

3. Flujo de CI/CD:

1. Un desarrollador realiza un cambio en el código y lo envía a GitHub.
2. Jenkins monitorea el repositorio de GitHub y desencadena una compilación cuando se detecta un cambio.
3. Jenkins realiza pruebas unitarias, pruebas de integración y análisis estáticos de código utilizando SonarQube.
4. Si todas las pruebas pasan con éxito y el análisis de código es aceptable, Jenkins despliega la aplicación en un entorno de pruebas.
5. Una vez aprobado en el entorno de pruebas, Jenkins despliega automáticamente la aplicación en un entorno de producción.
6. Ansible se encarga de la gestión de la configuración y el aprovisionamiento de instancias EC2 y otros recursos de AWS necesarios para el despliegue.

**Consideraciones Adicionales:**

* Se pueden implementar pipelines de CI/CD separados para cada servicio, permitiendo un despliegue independiente y ágil.
* Se deben establecer alarmas y métricas en AWS CloudWatch para monitorear el rendimiento de la aplicación y la infraestructura.
* Se deben establecer políticas de respaldo y recuperación en caso de fallos o pérdida de datos.
* Se debe considerar la implementación de seguridad a nivel de aplicación (como autenticación y autorización) para proteger los datos y garantizar el cumplimiento normativo.

Esta arquitectura proporciona una base sólida para desarrollar, desplegar y mantener un sistema de comercio electrónico escalable, seguro y de alta calidad utilizando tecnologías modernas y servicios en la nube de AWS.