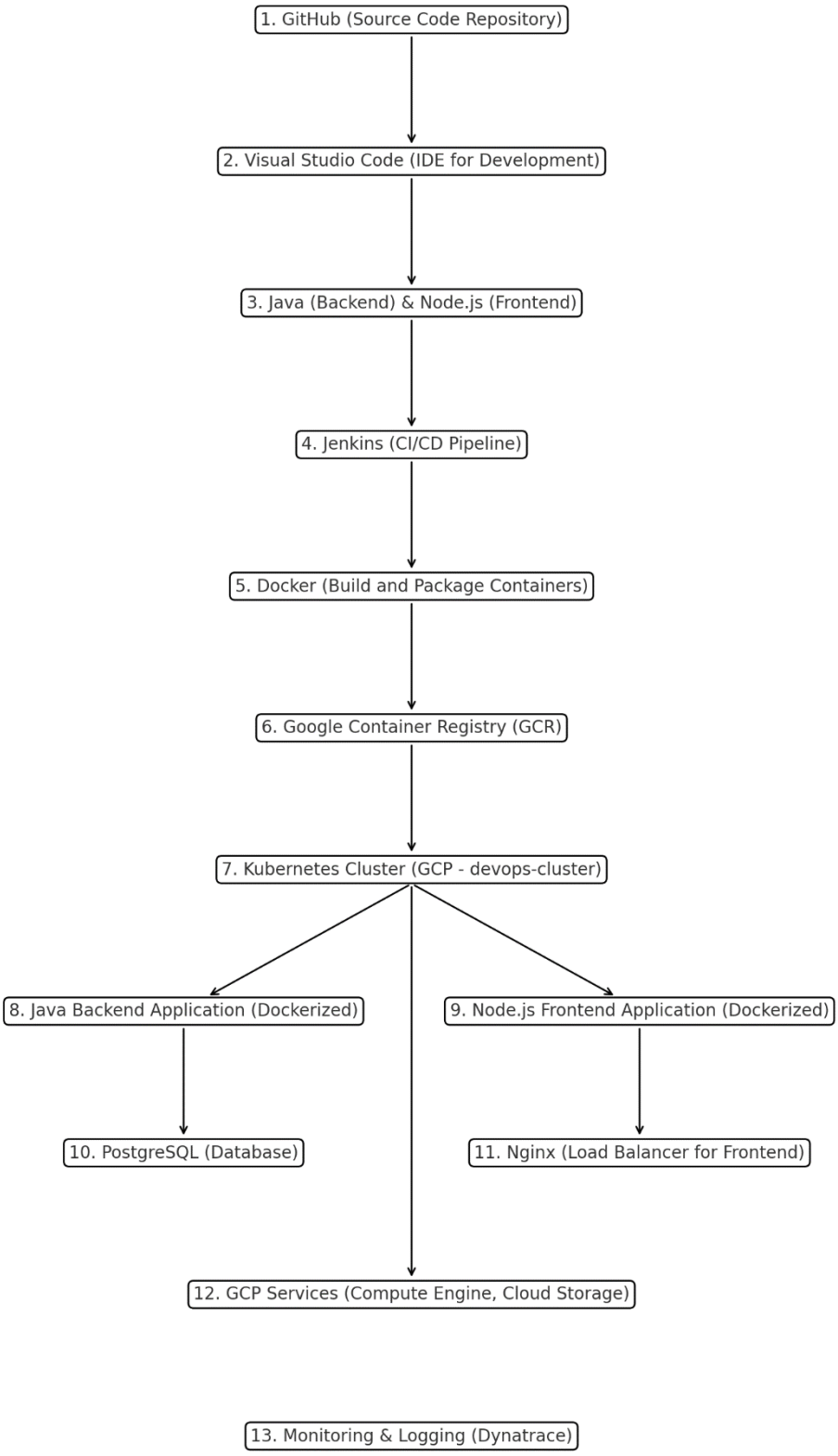
1. GitHub (Source Code Repository)
2. Visual Studio Code (IDE for Development)
3. Java (Backend) & Node.js (Frontend)
4. Jenkins (CI/CD Pipeline)
5. Docker (Build and Package Containers)
6. Google Container Registry (GCR)
7. Kubernetes Cluster (GCP - devops-cluster)
8. Java Backend Application (Dockerized)
9. Node.js Frontend Application (Dockerized)
10. PostgreSQL (Database)
11. Nginx (Load Balancer for Frontend)
12. GCP Services (Compute Engine, Cloud Storage)
13. Monitoring & Logging (Dynatrace)

Plan de Implementación del Pipeline CI/CD en Kubernetes (GCP)

Este documento describe el plan para implementar un pipeline CI/CD utilizando Jenkins y desplegando una aplicación backend en Java y frontend en Node.js en un clúster de Kubernetes en Google Cloud Platform (GCP). A continuación, se detallan los pasos y herramientas utilizadas para asegurar un despliegue exitoso de las aplicaciones.

Tecnologías y Herramientas Utilizadas

1. GitHub: Repositorio de código fuente donde se almacenará todo el código de la aplicación, incluyendo el backend en Java y el frontend en Node.js, así como los archivos de configuración de Kubernetes.
2. Visual Studio Code: Entorno de desarrollo (IDE) utilizado para escribir y gestionar el código del proyecto.
3. Java (Backend) y Node.js (Frontend): Lenguajes utilizados para desarrollar la aplicación. El backend maneja la lógica de negocio y la conexión a la base de datos, mientras que el frontend proporciona la interfaz de usuario.
4. Jenkins: Servidor de CI/CD que gestionará los pipelines para compilar, testear, y desplegar las aplicaciones en el clúster de Kubernetes.
5. Docker: Se utilizará para crear y empaquetar contenedores de las aplicaciones backend y frontend, asegurando que sean consistentes en cualquier entorno.
6. Google Container Registry (GCR): Servicio de GCP para almacenar las imágenes Docker que serán utilizadas en el despliegue de Kubernetes.
7. Kubernetes (GCP - devops-cluster): Clúster configurado en GCP que orquestará los contenedores de las aplicaciones.
8. PostgreSQL: Base de datos utilizada para almacenar los datos de la aplicación, alojada dentro del clúster de Kubernetes.

9. Nginx: Se utilizará como balanceador de carga para gestionar el tráfico hacia la aplicación frontend en Node.js.

10. DuckDNS: Servicio de DNS dinámico que se utilizará para configurar el Ingress, permitiendo el acceso externo a las aplicaciones en el clúster de Kubernetes.

Pasos Detallados para la Implementación

1. Desarrollo de la Aplicación: Utilizando Visual Studio Code, se desarrollará el backend en Java y el frontend en Node.js, siguiendo buenas prácticas de programación y aplicando tests unitarios para asegurar la calidad del código.

2. Configuración del Repositorio: El código fuente y los archivos de configuración se subirán a un repositorio GitHub, donde Jenkins tendrá acceso para gestionar el pipeline de CI/CD.

3. Configuración de Jenkins: Se configurará un pipeline en Jenkins que compilará y testeará el código, luego construirá las imágenes Docker de ambas aplicaciones (backend y frontend) y las subirá a Google Container Registry (GCR).

4. Despliegue en Kubernetes: Una vez que las imágenes Docker estén en GCR, el pipeline desplegará los contenedores en el clúster Kubernetes (GCP - devops-cluster). Los archivos YAML de configuración definirán los servicios y los deployments necesarios.

5. Configuración de Ingress con DuckDNS: Se configurará un Ingress en el clúster utilizando DuckDNS para asignar un dominio dinámico que permita el acceso externo a las aplicaciones. Esto incluye la configuración de reglas en el balanceador de carga para redirigir el tráfico adecuadamente entre el frontend y el backend.

6. Monitoreo y Log Management: Se integrará Dynatrace para monitorear el rendimiento de las aplicaciones y la infraestructura, asegurando que cualquier problema sea detectado y solucionado rápidamente.

Conclusión

Este plan de implementación detalla cada componente, herramienta y servicio utilizado en el proceso de CI/CD para asegurar una entrega continua y eficiente de las aplicaciones en un entorno Kubernetes gestionado en GCP.