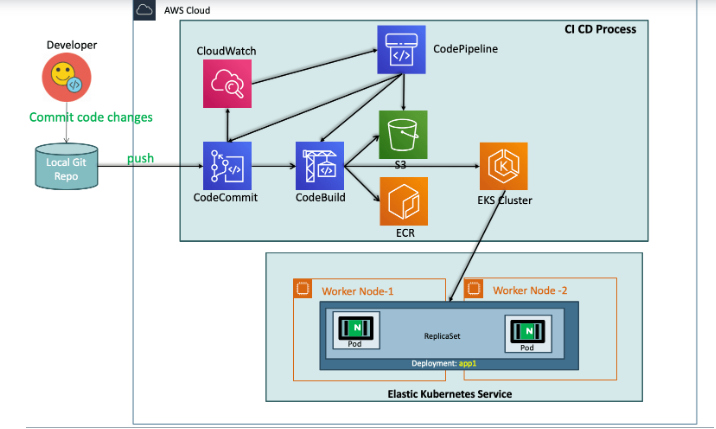
Creating a reference architecture for deploying a custom WooCommerce-based product on AWS using TypeScript CDK for three-tier application with container technology and AWS Pipeline involves several components. Below is a high-levelarchitecture diagram and guide the deployment process.



**Architecture Components:**

**Load Balancer:**

Distributes incoming traffic across multiple instances of Web Servers for high availability and scalability.

**Web Server:**

Hosts the WooCommerce application. Can be deployed using EC2 instances, ECS (Elastic Container Service), or Fargate depending on your requirements and preferences.

Configured to serve static content, handle HTTP requests, and interact with the App Server.

**App Server:**

Executes the business logic of the application.Deployed using ECS or Fargate for containerized execution.

Communicates with the Database Server to fetch/store data and processes dynamic requests from the Web Server.

**Database Server:**

Stores product information, user data, and other application-related data.

Utilizes RDS (Relational Database Service) or Aurora for managed database services.

Provides reliability, scalability, and security for data storage.

**Deployment Workflow:**

**Infrastructure Provisioning:**

Define infrastructure components using TypeScript CDK.

Instantiate Load Balancer, Web Server, App Server, and Database Server using CDK constructs.

Define networking configurations, security groups, and IAM roles as necessary.

**Containerization:**

Dockerize the Web Server and App Server components of the application.

Write Dockerfiles to encapsulate application dependencies and configurations.

Continuous Integration/Continuous Deployment (CI/CD):

Set up AWS CodePipeline for automating the build, test, and deployment process.

Configure CodePipeline stages for source code integration, containerization, and deployment to AWS infrastructure.

**Deployment Strategy:**

Use blue-green deployment or rolling deployment strategy for updating application versions.

Ensure zero-downtime deployment by gradually shifting traffic from the old version to the new version.

**Monitoring and Logging:**

Implement Amazon CloudWatch for monitoring application performance, resource utilization, and log management.

Set up alarms and notifications to respond to performance issues or system failures proactively.

**Security Considerations:**

Implement network security best practices such as VPC (Virtual Private Cloud) configuration, security groups, and NACLs (Network Access Control Lists).

Enable encryption at rest and in transit for data stored in the Database Server.

Implement IAM policies to control access to AWS resources and services.